IBM z/VSE
VSE Central Functions

VSE/POWER Application Programming

Version 9 Release 2

SC34-2642-01

IBM z/VSE
VSE Central Functions

VSE/POWER Application Programming

Version 9 Release 2

SC34-2642-01

Note!

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

This edition applies to Version 9 Release 2 of IBM VSE/POWER, which is part of VSE/Central Functions, Program Number 5686-CF9, and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC34-2642-00.

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

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Deutschland Research & Development GmbH
Department 3282
Schoenaicher Strasse 220
D-71032 Boeblingen
Federal Republic of Germany

You may also send your comments by FAX or via the Internet:

Internet: s390id@de.ibm.com
FAX (Germany): 07031-16-3456
FAX (other countries): (+49)+7031-16-3456

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

Contents

Figures

Tables

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785, U.S.A.

Any pointers in this publication to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement. IBM accepts no responsibility for the content or use of non-IBM websites specifically mentioned in this publication or accessed through an IBM website that is mentioned in this publication.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

```
IBM Deutschland GmbH
Dept. M358
IBM-Allee 1
71139 Ehningen
Germany
```

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

IPv6/VSE is a registered trademark of Barnard Software, Inc.

Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/VSE enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using Assistive Technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/VSE. Consult the assistive technology documentation for specific information when using such products to access z/VSE interfaces.

About This Publication

This publication is intended to give the reader guide and reference information for application programming related to IBM® VSE/POWER, the spooling component of z/VSE.

Who Should Use This Publication

This publication addresses application programmers who develop programs related to IBM VSE/POWER. The intended audience of this publication consists therefore of programmers who have familiarity in the following areas:

- User level knowledge of z/VSE
- Knowledge of IBM assembler language
- Basic knowledge of how to compile, debug and run assembler language programs.

How to Use This Publication

The following list tells where you find information on various aspects of VSE/POWER described in this publication:

- Chapter 1, "Understanding Syntax Diagrams," on page 3 describes how to read syntax diagrams.
- Chapter 2, "Job Accounting," on page 7 tells how to control job accounting.
- Chapter 3, "Output Segmentation," on page 31 describes output segmentation.
- Chapter 4, "Dynamic Access to VSE/POWER Job Attributes," on page 51 describes how to obtain the values of VSE/POWER job attributes dynamically in a program.
- Chapter 5, "Support of the IBM 4248 Printer," on page 53 describes the support of the IBM 4248 printer.
- Chapter 6, "Introduction to Spool-Access Support," on page 57 describes the spool-access support of VSE/POWER; it allows a program running under or outside the control of VSE/POWER to access the services of VSE/POWER.
- Chapter 7, "CTL - Passing a Command," on page 65 describes how to send a VSE/POWER command via the spool-access support to VSE/POWER and how to retrieve the resulting information.
- Chapter 8, "GET - Retrieving a Queue Entry," on page 75 shows how to retrieve an entry from a VSE/POWER queue.
- Chapter 9, "PUT - Submitting a Job, a Job Stream, or Output," on page 103 describes how to submit a queue entry to a VSE/POWER queue.
- Chapter 10, "GCM - Retrieving Job Event and Output Generation Messages, Inquiring eXtended Event Messages," on page 139 describes how user-written application programs can retrieve job event and output generation messages for jobs which have been submitted to VSE/POWER and can inquire messages which inform about creation/alteration/deletion events in VSE/POWER queues (eXtended Event Messages).
- Chapter 11, "Supporting I/O Devices Via Device Driving Systems," on page 171 describes how I/O devices may be used which are not directly supported by VSE/POWER.

- Chapter 12, “Spool-Access Support Macros,” on page 211 lists and describes the macros to be used with the spool-access support.
- Chapter 13, “Spool-Access Support Programming Example,” on page 271 gives an example of how the different spool-access support services may be applied in a programming environment.
- Chapter 14, “Return and Feedback Codes and Their Meanings,” on page 297 lists all possible return and feedback codes of the spool-access support.
- Chapter 15, “Writing Various Exit Routines,” on page 317 tells how to write exit routines for the customized handling of local input and output.

The following information is included in the appendix:
- Appendix A, “Cross-Partition Communication via Spool Macros,” on page 343 describes the XECB-macro based cross-partition communication support.
- Appendix B, “Output Segmentation by SEGMENT Macro,” on page 371 describes the use of the SEGMENT macro.
- Appendix C, “Spool-Access Support Graphical Description,” on page 379 describes the spool-access support using graphical representation.

Additional help is provided at the back of the publication:
- The glossary explains technical terms.
- The index helps you to locate information.

Where to Find More Information

The following IBM publications also describe aspects of VSE/POWER:
- *VSE/POWER Administration and Operation*, SC34-2625
- *VSE/POWER Remote Job Entry*, SC33-6734
- *VSE/POWER Networking*, SC34-2603

The VSE/POWER messages are listed in *z/VSE Messages and Codes, Volume 1*, SC34-2632.

For z/VSE, you may occasionally need the following IBM publications:
- *z/VSE Installation*, SC34-2631
- *z/VSE Operation*, SC33-8309
- *z/VSE SNA Networking Support*, SC34-2626
- *z/VSE Guide to System Functions*, SC33-8312
- *z/VSE System Control Statements*, SC34-2637
- *z/VSE System Macros User’s Guide*, SC33-8407
- *z/VSE System Macros Reference*, SC34-2638

For information on VTAM, see
- *Planning for NetView, NCP, and VTAM*, SC31-8063

z/VSE Home Page

z/VSE has a home page on the World Wide Web, which offers up-to-date information about VSE-related products and services, new z/VSE functions, and other items of interest to VSE users.

You can find the z/VSE home page at

http://www.ibm.com/systems/z/os/zvse/

You can also find VSE User Examples (in zipped format) at

http://www.ibm.com/systems/z/os/zvse/downloads/samples.html

VSE/POWER Web Page

You can find current information on VSE/POWER at

http://www.ibm.com/systems/z/os/zvse/products/cf.html#power

Abbreviations

```
ACB      =  access method control block
ACF      =  advanced communication function
ASA      =  records with American National Standard control characters
BAM      =  Basic Access Method
BMS      =  basic mapping support (used by CICS)
BSC      =  binary synchronous communication
BSM      =  Basic Security Manager
CAT      =  common address table
CCB      =  channel control block
CCW      =  channel control word
CICS/VSE =  Customer Information Control System/VSE
CKD      =  count-key-data (disk device type)
CPDS     =  composed page data stream (also 'all-point addressable records')
DBLK     =  data block
DDS      =  device driving system
DSHR     =  data set header record
EBCDIC   =  extended binary-coded decimal interchange code
ECB      =  event control block
ESM      =  External Security Manager
FBA      =  fixed-block architecture (disk format)
FCB      =  forms control buffer (for printer control)
GCM      =  get completion message
ID       =  identifier
JCL      =  job control language
JECL     =  job entry control language
KB       =  Kilobyte (=1024 bytes)
MB       =  Megabyte (=1024 KB)
MCC      =  magnetic card code
OPTB     =  output parameter text block
PNET     =  VSE/Power networking
PSF      =  Print Services Facility*
RJE      =  Remote Job Entry
SAS      =  Spool-Access Support
SCS      =  standard character string
SNA      =  system network architecture
SPL      =  spool parameter list
SVA      =  system virtual area
SVC      =  supervisor call
TCB      =  task control block
TCP/IP   =  Transmission Control Protocol/Internet Protocol
VIO      =  Virtual I/O storage space (used for queue file copy)
UCB      =  universal character set buffer
VM       =  Virtual Machine (a type of IBM operating systems)
VSE      =  Virtual Storage Extended (a type of IBM operating systems)
VSE/ESA  =  Virtual Storage Extended/Enterprise Systems Architecture
VTAM     =  Virtual Telecommunications Access Method
z/OS     =  zSeries eServer operating system
z/VM     =  zSeries VM
z/VSE    =  zSeries VSE
```

* Throughout this publication - if not stated otherwise - information given for the IBM 3800 Printing Subsystem applies also to the IBM 3200 Printing Subsystem.

Summary of Changes

This publication has been updated to reflect the enhancements and changes that are implemented with z/VSE Version 5 Release 2. It also includes terminology, maintenance, and editorial changes.

Summaries of changes for Version 3 Release 1 and older versions of z/VSE can be found in the *VSE/POWER Application Programming* for z/VSE Version 3 Release 1.

VSE/POWER 9.2

XEM Support for SAS interface

As of VSE/POWER 9.2, a new XEM (stands for eXtended Event Message) support has been introduced as an extension of the JCM/JGM/OGM support for a SAS user. XEM provides an application program with the opportunity to observe all important events in VSE/POWER queues.

VSE/POWER generates a fixed format 1Q5XI extended event message for a requesting application in the following cases:

- A new queue entry has been created within a VSE/POWER queue or spooled to a tape.
- An existing queue entry has been altered in a VSE/POWER queue.
- An existing queue entry has been deleted from a VSE/POWER queue (moved into the DEL queue).

As opposed to JCMs/JGMs/OGMs, which are created on request by a job being executed, creation of extended event messages requires that the XEM service has been started by an SAS application. For each application, VSE/POWER sets up a separate queue for keeping XEMs. An application requests messages from its own queue, retrieved messages are deleted from this queue after being sent.

Each application queue has a fixed number of message slots - 2048, one slot for one extended event message. Up to 32 applications can use XEM support concurrently. XEM support is based on the extended GCM service: GCM requests are used to initialize messages building/queuing and to retrieve messages as well.

For details, refer to “GCM-XEM Service” on page 155.

VSE/POWER 9.1

IPWSEGM Supports Duplicates for LST and PUN Output

Since z/VSE 4.1, VSE/POWER supports the creation of duplicate output using the * $$ LSTDUP and * $$ PUNDUP JECL statements or the PCOPY command. Output duplication allows multiple VSE/POWER tasks to access a single image of spooled data. Output duplication has now been made available for program-driven segmentation via IPWSEGM. Duplication for the next output segment can be requested using new operand DUP=YES for statements * $$ LST or * $$ PUN supplied via macro IPWSEGM. For each duplicate, the supplied JECL must contain DUP=YES followed by at least one JECL operand permitted for statements * $$

LSTDUP or * $$ PUNDUP. For additional details, refer to “Generation of Duplicate LST and PUN Output” on page 32.

Enhanced Dynamic Access to VSE/POWER Job Attributes

A VSE/POWER job can create multiple LST and PUN outputs, each with a different job name and other properties. From z/VSE 5.1 onwards, a common attribute TKN has been defined for each job and all of its spooled output. The TKN attribute of the VSE/POWER job can now be extracted from the MAPPOWJB DSECT using the GETFLD FIELD=POWJOB service. See Chapter 4, “Dynamic Access to VSE/POWER Job Attributes,” on page 51.

VSE/POWER 8.3

OGM Support for SAS Interface

Prior to version 8.3, VSE/POWER did not provide notifications for SAS user about outputs produced by jobs, like it does for job generation and job completion events by issuing 1Q5HI and 1Q5DI messages. Starting with version 8.3, VSE/POWER issues a new fixed format informational message, 1Q5RI, for notification about output generation event. This 1Q5RI message is generated and issued:

- When a job, submitted via SAS interface, which creates an LST or PUN entry (or XMT entry if LST/PUN output is designated for sending to another PNET node) and this entry is ready for processing.
- When a job has been submitted by a PUT request with new options specified in SPL.

New 1Q5RI message is processed similarly as the existing 1Q5HI and 1Q5DI messages, which are placed into the SAS messages queues (user queue, common queue, or both), and can be retrieved by a GCM request later on.

To handle the increased number of issued fixed format messages, the default queue size of fixed format messages has been increased from 20 to 50, and its maximal value from 99 to 255.

For additional details see Chapter 9, “PUT - Submitting a Job, a Job Stream, or Output,” on page 103 and Chapter 10, “GCM - Retrieving Job Event and Output Generation Messages, Inquiring eXtended Event Messages,” on page 139.

Enhancement of Selection Criteria for SAS GCM Interface

So far using selection criteria of SAS GCM you could retrieve:

- All event messages
- All messages from jobs with a specific job name
- All messages from job with a specific job name and job number

Starting with VSE/POWER 8.3, you can retrieve additionally:

- All messages of specific type (JCM, JGM or OGM)
- All messages of specific type from jobs with a specific job name
- All messages of specific type from jobs with a specific job name and job number

See “Message Selection Criteria” on page 144.

Part 1. Syntax Diagrams, Accounting, Output Segmentation, Dynamic Access to Job Attributes, and IBM 4248 Printer Support

Chapter 1. Understanding Syntax Diagrams

This section describes how to read the syntax diagrams in this publication.

To read a syntax diagram follow the path of the line. Read from left to right and top to bottom.

- The ▶▶─── symbol indicates the beginning of a syntax diagram.
- The ───▶ symbol, at the end of a line, indicates that the syntax diagram continues on the next line.
- The ▶─── symbol, at the beginning of a line, indicates that a syntax diagram continues from the previous line.
- The ───▶◀ symbol indicates the end of a syntax diagram.

Syntax items (for example, a keyword or variable) may be:

- Directly on the line (required)
- Above the line (default)
- Below the line (optional)

Uppercase Letters

Uppercase letters denote the shortest possible abbreviation. If an item appears entirely in uppercase letters, it can not be abbreviated.

You can type the item in uppercase letters, lowercase letters, or any combination. For example:

```
▶▶──KEYWOrd──────────────────────────────────────────────────────────▶◀
```

In this example, you can enter KEYWO, KEYWOR, or KEYWORD in any combination of uppercase and lowercase letters.

Symbols

You **must** code these symbols exactly as they appear in the syntax diagram

*****	Asterisk
:	Colon
,	Comma
=	Equal Sign
-	Hyphen
//	Double slash
()	Parenthesis
.	Period
+	Add

For example:

```
* $$ LST
```

Variables

An *italicized* lower-case word indicates a variable that you must substitute with specific information. For example:

```
►►──,USER=user_id──────────────────────────────────────────────►◄
```

Here you must code USER= as shown and supply an ID for user_id. You may, of course, enter USER in lowercase, but you must not change it otherwise.

Repetition

An arrow returning to the left means that the item can be repeated.

```
     ┌───────┐
     ▼       │
►►───repeat──┴─────────────────────────────────────────────────►◄
```

A character within the arrow means you must separate repeated items with that character.

```
     ┌─,─────┐
     ▼       │
►►───repeat──┴─────────────────────────────────────────────────►◄
```

A footnote (1) by the arrow references a limit that tells how many times the item can be repeated.

```
     ┌────────────┐
     ▼  (1)       │
►►────────repeat──┴────────────────────────────────────────────►◄
```

Notes:

1 Specify *repeat* up to 5 times.

Defaults

Defaults are above the line. The system uses the default unless you override it. You can override the default by coding an option from the stack below the line. For example:

```
     ┌─A─┐
►►───┤   ├─────────────────────────────────────────────────────►◄
     ├─B─┤
     └─C─┘
```

In this example, A is the default. You can override A by choosing B or C.

Required Choices

When two or more items are in a stack and one of them is on the line, you **must** specify one item. For example:

```
►►───┬─A─┬─────────────────────────────────────────────────────►◄
     ├─B─┤
     └─C─┘
```

Here you must enter either A or B or C.

Optional Choice

When an item is below the line, the item is optional. Only one item **may** be chosen. For example:

```
►►──┬───┬──────────────────────────────────────►◄
    ├─A─┤
    ├─B─┤
    └─C─┘
```

Here you may enter either `A` or `B` or `C`, or you may omit the field.

Required Blank Space

A required blank space is indicated as such in the notation. For example:

```
* $$ EOJ
```

This indicates that at least one blank is required before and after the characters $$.

Chapter 2. Job Accounting

Job Accounting by VSE/POWER

Requirements

To have VSE/POWER job accounting support available, the system programmer specifies the account records that are needed in the POWER generation macro by either ACCOUNT=YES (all types wanted) or by, for example, ACCOUNT=(AFP,BSC,..,XSPOOL) for selected types. VSE/POWER needs some additional processor, virtual storage, and disk space to accommodate the VSE/POWER account file. For information about these requirements, see *VSE/POWER Administration and Operation*, SC34-2625.

Account Macros and Records

If accounting support is available, VSE/POWER automatically collects job accounting information for every partition under its control and stores this information for every job step in chronological order in the account file. In this file, and also on tape or disk if the file was saved, the records are stored sequentially:

```
On CKD disk or tape:  in variable unblocked format
On FBA disk:          in variable blocked format
```

VSE/POWER produces various types of account records, and you can write an evaluation program of your own to process these records.

Moreover, you may want to write a program under the name of $JOBACCT to add information to the execution account record specially.

Available Account Macros

The PACCNT macro, to be used in your processing program, requests a DSECT to be assembled into your program for any, several, or all types of account records.

The PUTACCT macro lets you add information to the execution save account record.

Syntax rules

For an explanation of the syntax used in the formats of these macros, see Chapter 1, “Understanding Syntax Diagrams,” on page 3. Continuation codes that may be required in column 72 are not shown as part of the macro formats.

Available Account Records

The following types of account records are supported. (The account-record ID is in position 43 of every record).

Table 1. VSE/POWER Account Record Overview

Account Record Type	ID	Figure/Page
Advanced Function Printing account record	A	Table 4 on page 11
Execution account record	E	Table 5 on page 13

Table 1. VSE/POWER Account Record Overview (continued)

Account Record Type	ID	Figure/Page
List account record	L	Table 6 on page 15
Network account record	N	Table 7 on page 17
Punch account record	P	Table 8 on page 19
Reader account record	R	Table 9 on page 21
Transmitter account record	M	Table 10 on page 22
Receiver account record	V	Table 10 on page 22
RJE,BSC account record	T	Table 11 on page 23
RJE,SNA account record	S	Table 12 on page 24
System-up account record	U	Table 13 on page 24
Spool-access-connect account record	C	Table 14 on page 25
Spool-access-operation account record	X	Table 15 on page 26

Account-File-Full Condition

If the account file is full and a task of VSE/POWER must write another account record, this task waits until the operator issues a PACCOUNT command. Instruct your operator to do one of the following:

- Save the account records on tape (generally the preferred action). For their format, see "Layout of the Execution Account Record" on page 12 and the following sections, but note that every record, in addition, has the standard 8-byte prefix (BAM) for variable length records.
- Save the account records on disk if a disk extent has been defined for this purpose. For their format, see "Layout of the Execution Account Record" on page 12 and the following sections, but note that every record, in addition, has the standard prefix for variable length records.
- Have the contents of the account file spooled to the punch queue and punched out by starting a punch-writer task with class P.

  When VSE/POWER is to spool the account file's contents, every punch record has the format shown in Table 2.

Table 2. Account File Record Format When Spooled to PUNCH Queue

Columns	Contents
1	Account-record ID (field ACIDEN of the account record)
2-72	Data (bytes 0-70 of the account record) punched in the same positions as it appears in the account record, in- cluding the account record ID (invalid for continuation cards).
73-78	Record number of the account file.
79-80	Sequence number of continuation cards. One account record may require one or more punched cards.

Record Format With or Without Prefix

In a shared spooling environment, (more precisely: as soon as the VSE/POWER macro specifies the SYSID= operand or the SET SYSID= autostart statement is used), the following 16 bytes are placed at the beginning of the data bytes of each generated account record. The SYSID operand of the PACCNT macro must be used in this case to reflect the presence of the shared header.

Table 3. Account-Record Prefix for Systems with SYSID Only

Field Name	Description	Field Type & Length
ACSYSID	System ID	CL1
ACTYPE	Account Record Type (character) and version X'09'	CL2
ACCOMP	Component ID: '5686CF9'	CL8
	Reserved	CL5

The PACCNT Macro: Generating an Account-Record DSECT

PACCNT requests a DSECT to be assembled into your program for one, several, or all types of account records. For most account-record types, the generated DSECT includes a fixed (header) part and a variable part. In your program, examine the account record ID and then work with the labels that apply to the specific record type. Read the generated DSECT before you start coding a program using it.

Format of the Macro

The format of the macro as shown below does not include the continuation character, which you may have to code in column 72.

```
                          ┌─,──────────┐
►►──┬──────┬── PACCNT─────▼────────────┴──────────────────────────►◄
    └─name─┘               ├─ALL=YES────┤
                           ├─AFP=YES────┤
                           ├─BSC=YES────┤
                           ├─EXEC=YES───┤
                           ├─LIST=YES───┤
                           ├─PNET=YES───┤
                           ├─PUNCH=YES──┤
                           ├─READER=YES─┤
                           ├─RECV=YES───┤
                           ├─SNA=YES────┤
                           ├─SYS=YES────┤
                           ├─SYSID=YES──┤
                           ├─TRANS=YES──┤
                           ├─XCONN=YES──┤
                           └─XSPOOL=YES─┘
```

For *name* (in the name field), specify the label you want to use for referring to the DSECT for the account record(s).

ALL=NO|YES
Specify ALL=YES to have DSECTs generated for all account records. If the

account prefix is required, then specify also SYSID=YES. Any other specification that you may supply is ignored.

AFP=NO|YES
Specify AFP=YES to have a DSECT generated for the truncated part of an Advanced Function Printing account record (for layout, see Table 4 on page 11).

BSC=NO|YES
Specify BSC=YES to have a DSECT generated for an RJE,BSC account record (for layout, see Table 11 on page 23).

EXEC=NO|YES
Specify EXEC=YES to have a DSECT generated for an execution account record and its extensions (for layout, see Table 5 on page 13).

LIST=NO|YES
Specify LIST=YES to have a DSECT generated for a list account record (for layout, see Table 6 on page 15).

PNET=NO|YES
Specify PNET=YES to have a DSECT generated for a PNET network account record (for layout, see Table 7 on page 17).

PUNCH=NO|YES
Specify PUNCH=YES to have a DSECT generated for a punch account record (for layout, see Table 8 on page 19).

READER=NO|YES
Specify READER=YES to have a DSECT generated for a reader account record (for layout, see Table 9 on page 21).

RECV=NO|YES
Specify RECV=YES to have a DSECT generated for a transmitter/receiver account record (for layout, see Table 10 on page 22).

SNA=NO|YES
Specify SNA=YES to have a DSECT generated for an RJE,SNA account record (for layout, see Table 12 on page 24).

SYS=NO|YES
Specify SYS=YES to have a DSECT generated for a system-up account record (for layout, see Table 13 on page 24).

SYSID=NO|YES
Specify SYSID=YES to have the 16-byte account-record prefix generated into and at the beginning of the account-record DSECT (for layout, and requirement, see Table 3 on page 9).

TRANS=NO|YES
Specify TRANS=YES to have a DSECT generated for a transmitter/receiver account record (for layout, see Table 10 on page 22).

XCONN=NO|YES
Specify XCONN=YES to have a DSECT generated for a spool-access connect account record (for layout, see Table 14 on page 25).

XSPOOL=NO|YES
Specify XSPOOL=YES to have DSECTs generated for a spool-access operation account record (for layout, see Table 15 on page 26).

Layout of the Advanced Function Printing (AFP) Account Record

VSE/POWER writes an Advanced Function Printing account record for a device driving system (DDS) whenever the DDS sends an 'account record order' which contains a valid AFP account record. This DDS is responsible for providing the contents of the standard VSE/POWER account record header and for defining the DDS specific layout of the record part starting at location ACAFPBBY. For the account record to be correct and unique, VSE/POWER updates fields ACIDEN, ACAFPLEN, and ACAPPLID before it writes the AFP record to the account file. For creation of an account record order see also section “Subsystem Orders” on page 204.

Independent of a specific device driving subsystem, the beginning of every Advanced Function Printing account record looks as follows:

Table 4. Advanced Function Printing Account Record

Field Name	Description	Field Type & Length
ACDATE	Date in the format as specified at SYSGEN; provided by the DDS	CL8
ACSTRT	Print start time (0HHMMSSF, where F = sign) packed decimal; provided by the DDS	PL4
ACSTOP	Print stop time (0HHMMSSF, where F = sign) packed decimal; provided by the DDS	PL4
ACUSER	16 bytes of user information; provided by the DDS from SPLDUI	CL16
ACNAME	Name of report (jobname); provided by the DDS from SPLGJB	CL8
ACNUMB	Job number as assigned by VSE/POWER; provided by the DDS from SPLGJN	BL2
ACIDEN	Record ID (A), set by VSE/POWER	CL1
ACCANC	Printer cancel code; provided by the DDS according to the VSE/POWER cancel codes of the Spool-Access-Operation Account Record	BL1
ACFLG1	Account flag byte 1: ACF1CE20 X'80' ON = ACDATE is 20yy OFF = ACDATE is 19yy	BL1
end of header		

Table 4. Advanced Function Printing Account Record (continued)

Field Name	Description	Field Type & Length
ACAFPLEN	Total length of AFP account record, set by VSE/POWER (optional 16 bytes SYSID prefix excluded)	BL2
ACAPPLID	XPCC application identifier of DDS set by VSE/POWER	CL8
ACAFPBDY	Start of specific Advanced Function Printing account information with layout as provided by corresponding device driving system (DDS). Length (x) of this section is the length provided in the field ACAFPLEN minus 45 bytes for the header and minus 10 bytes for the subsequent length and application-id fields.	CLx

Layout of the Execution Account Record

VSE/POWER builds one *execution account record* for every z/VSE job step (every time a // EXEC or /& statement occurs). If a job or job step is canceled, VSE/POWER's statistics reflect the processing up to the point of this cancelation. The record can be up to 2008 bytes long.The same execution account record is used by VSE/POWER itself when the operator has issued a PEND command. In this VSE/POWER execution account record, certain fields are set to zero (see below), and field EXDUSER contains the constant "VSE/POWER-E.A.R."

Note: The page count is reflected in both fields EXNPG (2 bytes) and EXPCNT (4 bytes). However, when EXNPG overflows, its value remains 65,535 permanently.

Table 5. Execution Account Record

Field Name	Description	Field Type & Length
ACDATE	Processing date in the format as defined for the system	CL8
ACSTRT	Start time of job step (0hhmmssf, where f = sign)	PL4
ACSTOP	Stop time of job step (0hhmmssf, where f = sign) This time may be higher than the time logged on the console; it accounts for VSE/POWER job termination	PL4
ACUSER	16 bytes of user information from * $$ JOB card, "VSE/POWER-E.A.R." for VSE/POWER exec. acct. record	CL16
ACNAME	Current VSE/POWER job name or AUTONAME "POWER/VS" for VSE/POWER exec. acct. record	CL8
ACNUMB	Job number assigned by VSE/POWER	BL2
ACIDEN	Record ID (E)VSE/POWER	CL1
ACCANC	cancel code: X'10' = Normal end of VSE/POWER job or task. The associated z/VSE job(s) may have been canceled by the system nevertheless. X'20' = PCANCEL was issued. X'30' = PSTOP command was issued. The code is not stored in the account record if the EOJ option was specified in the PSTOP command. X'40' = PFLUSH command was issued. X'70' = The job was canceled due to an I/O error.	BL1
ACFLG1	Account flag byte 1: ACF1CE20 X'80' ON = ACDATE is 20yy OFF = ACDATE is 19yy	BL1
end of header		
	Reserved	CL3
EXFRM	FROM remote ID	BL1
	Reserved	BL1
EXICL	Class	CL1
EXIPR	Priority	CL1
EXNLN	Number of lines spooled Binary zero for VSE/POWER exec. acct. record	BL4
EXNCD	Number of cards spooled Binary zero for VSE/POWER exec. acct. record	BL4
EXNPG	Number of pages spooled (the value in this field is limited to 65,535 pages; if dealing with larger numbers, use field EXPCNT instead). Binary zero for VSE/POWER exec. acct. record	BL2
EXSIO	Length of SIO table (including the byte containing X'20')	BL2
EXTAC	Length of total execution account record	BL2
EXPOWEX	Offset to VSE/POWER extension area from ACDATE	BL2
EXUSREX	Offset to user PUTACCT extension area from ACDATE	BL2
EXOJ#	Original job number, if one exists	BL2
EXXNODE	Name of execution node	CL8
EXFRNO	Name of FROM (originating) node	CL8
EXFRUS	ID of originating user	CL8

Table 5. Execution Account Record (continued)

Field Name	Description	Field Type & Length
EXDJOB	z/VSE job name from the // JOB card	CL8
EXDUSER	16 bytes of user information from the // JOB card On VSE/POWER shutdown, the field contains: "VSE/POWER-E.A.R."[1]	CL16
EXPID	Partition ID in EBCDIC formatz/VSE	CL2
EXDCANC	cancel code (refer to *z/VSE Messages and Codes*)	BL1
EXTYPE	Type of record; S = job step L = Last step	CL1
EXJDUR	Duration of job step (in 300ths of a second)	BL4
EXPHASE	Phase name, taken from the // EXEC card	CL8
EXPASZ	Number of pages multiplied by page size in KB	BL4
EXCPUTM	Processor time in 300ths of a second. This is the actual time used by a job or job step in the system.	BL4
EXOVHTM	Overhead time in 300ths of a second. This is the time needed for activities that cannot be charged to a specific program or partition. For example, the time for calling a routine, for error recovery, or from the start of the $JOBACCT routine to the processing of the // EXEC statement. All SVC processing is counted as active processor time for the job or job step. Overhead time is distributed over active partitions in proportion to used CPU time	BL4
EXALLTM	Total system wait time in 300ths of a second. All bound time is distributed in equal parts between all active partitions.	BL4
EXSIOTB	SIO tables. Six bytes per device defined to the system during system startup provided that at least one I/O request has been performed for the device: bytes 0 and 1 = 0cuu; bytes 2 through 5 = count of SIOs in current job step. Note that VSE/POWER may suppress SIO table entries if the maximum account record length of 2008 bytes is exceeded.	BL6 for every SIO entry
EXSIOTB + n - 1	Set by the system to X'20'. n = total length of the SIO tables (EXSI0)	BL1
EXSIOTB+n	User PUTACCT extension area, also to be found by the pointer in field EXUSREX above.	undef.
EXPACCT	Start of the VSE/POWER extension area, also to be found by the pointer in field EXPOWEX above. Starts by the network account number.	CL8
EXPCNT	Total number of pages spooled	BL4
	Reserved	CL20

[1] E.A.R. stands for 'execution account record'.

Addressing Scheme of the Execution Account Record

The following graphic (starting at field ACDATE) shows how the two fields EXPOWEX and EXUSREX point to the respective beginnings of the VSE/POWER and of the user extension areas in the execution account record.

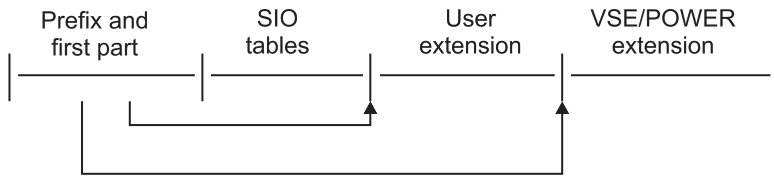


Figure 1. Addressing Scheme within the Execution Account Record

Layout of the List Account Record

VSE/POWER builds a *list account record* for every queue entry that is processed by a list task.

Note: The page count is reflected in the two byte fields LSTPAG/LSTEXP and, at the same time, in the four byte fields LSTPGN/LSTEPGN. When the two byte fields overflow, they will permanently contain the value of 65,535.

Table 6. List Account Record

Field Name	Description	Field Type & Length
ACDATE	Processing date in the format as defined to the system	CL8
ACSTRT	Start time of list output (0hhmmssf, where f = sign)	PL4
ACSTOP	Stop time of list output (0hhmmssf, where f = sign)	PL4
ACUSER	16 bytes of user information from the * $$ JOB or the * $$ LST statement	CL16
ACNAME	VSE/POWER job name from the * $$ JOB or the // JOB statement	CL8
ACNUMB	Job number assigned by VSE/POWER (same as that of the associated reader queue entry for first * $$ LST statement)	BL2
ACIDEN	Record ID (L)	CL1
ACCANC	VSE/POWER cancel code: X'10' = Normal end of VSE/POWER job or task - The associated z/VSE job(s) may have been canceled by the system nevertheless. X'30' = PSTOP command was issued. The code is not stored in the account record if the EOJ option was specified in the PSTOP command. X'40' = PFLUSH command was issued. X'70' = The job was canceled due to an I/O error.	BL1
ACFLG1	Account flag byte 1: ACF1CE20 X'80' ON = ACDATE is 20yy OFF = ACDATE is 19yy	BL1
end of header		

Table 6. List Account Record (continued)

Field Name	Description	Field Type & Length
LSTADR	Printer or RJE-line address (cuu), SNA, or GSP	CL3
LSTFRM	FROM remote ID	BL1
LSTTO	TO remote ID	BL1
LSTOCL	Printed output class	CL1
LSTOPR	Printed output priority number	CL1
LSTNUM	No. of lines printed only for 1st copy,comprising: - lines of total entry if ACCANC=X'10', or - lines of partial entry if ACCANC=X'30/40/70' Note the additional lines in LSTEXR	BL4
LSTTRK	Number of DBLK groups for output storage The field is set to zero if the list output was spooled to tape.	BL2
LSTSUF	Job suffix (segment) number assigned by VSE/POWER If: X'00' - The only segment for a job X'82' or higher - The last segment for a job; the seven low-order bits give the number of segments	BL1
LSTCOP	Number of complete printed copies. (If more than one copy, statistics are totals for all copies)	BL1
LSTFOR	Print-forms ID	CL4
LSTEXR	No. of extra lines printed beyond LSTNUM due to - PRESTART or PSETUP command request - separator pages - copies 2,3,..,n if ACCANC=X'10' or - copies 2,3,..,x if ACCANC=X'30/40/70' where "x" is the partially printed copy	BL4
LSTPAG	No.of pages printed only for 1st copy comprising: - pages of total entry if ACCANC=X'10', or - pages of partial entry if ACCANC=X'30/40/70' Note the additional pages (skip-chan-1 or filled pages due to FCB or LTAB) in LSTEXP (The value in this field is limited to 65,535 pages; if dealing with larger numbers, use field LSTPGN instead)	BL2
LSTEXP	No. of extra pages printed additionally to LSTPAG due to - PRESTART or PSETUP command request - separator pages - copies 2,3,..,n if ACCANC=X'10' or - copies 2,3,..,x if ACCANC=X'30/40/70' where "x" is the partially printed copy (The value in this field is limited to 65,535 pages; if dealing with larger numbers, use field LSTEPGN instead)	BL2
LSTFLSH	Flash ID (applies only to 3800 printer)	
LSTCPYG	Copy groupings (applies only to 3800 printer)	CL4
LSTNODE	Name of own node (system) in the network	CL8
LSTTOUS	Destination-user (TO) identification	CL8
		CL8

Table 6. List Account Record (continued)

Field Name	Description	Field Type & Length
LSTFRNO	Name of originating (FROM) node	CL8
LSTFRUS	Originating-user (FROM) identification	CL8
LSTOJ#	Original job number, if one exists	BL2
LSTACCT	Network Account number	CL8
	Reserved	CL2
LSTRINS	Records inserted by OUTEXIT routine	BL4
LSTRDEL	Records deleted by OUTEXIT routine	BL4
LSTPGN	Number of pages printed	BL4
LSTEPGN	Number of extra pages printed	BL4
LSTPGRNM	Programmer's name	CL20
LSTBLDG#	Programmer's building number	CL8
LSTROOM#	Programmer's room number	CL8
LSTDEPT#	Programmer's department number	CL8
LSTDIST	Distribution code	CL8

Layout of the Network Account Record

VSE/POWER builds a *network account record* for an existing communication path when a session via this path is terminated. The record contains information about all activities during the session.

Table 7. Network Account Record

Field Name	Description	Field Type & Length
NETDTE	Processing date in the format as defined for the system	CL8
NETSGN	Sign-on time (0hhmmssf, where f = sign)	PL4
NETSGF	Sign-off time (0hhmmssf, where f = sign)	PL4
NETNODE	ID of the connected node	CL8
NETNPAS	Node password	CL8
NETPSW	Line password	CL8
NETICNT	Invalid responses per session	BL2
NETIDEN	Record ID (N)	CL1
NETTERM	Signoff code and century indicator: X'01' = ON = NETSOD is 20yy OFF = NETSOD is 19yy X'02' = Normal VTAM shutdown X'04' = Abnormal end of VTAM X'08' = An internal error occurred X'10' = A line error occurred or a session was terminated X'20' = A time-out occurred X'40' = A remote SIGNOFF occurred X'80' = Cancel on operator request (one of the commands PSTOP and PEND)	BL1
NETTEC	Terminal error count	BL1
end of header		

Table 7. Network Account Record (continued)

Field Name	Description	Field Type & Length
NETLAD	Line address or 'SNA' or 'TCP' or 'SSL'	CL3
NETTRAN	NETTRAN transmission count (of buffers) per session For PNET support using BSC and CTC: - number of started I/O's or - number of sent buffers or - number of received buffers For PNET support using TCP or SSL: - number of CTC-simulated started I/O's For PNET support using SNA: - number of sent buffers	BL4
	For PNET support using BSC	
NETTCNT	Time-out count per session	BL2
NETERR	Error count per session	BL2
	For PNET support using SNA	
NETRCVE	Buffers received during session	BL4
NETSOD	Sign-off date	CL8

Layout of the Punch Account Record

VSE/POWER builds a *punch account record* for every punch-queue entry that is processed by a punch task.

Table 8. Punch Account Record

Field Name	Description	Field Type & Length
ACDATE	Processing date in the format defined to the system	CL8
ACSTRT	Start time of punch output (0hhmmssf, where f = sign).	PL4
ACSTOP	Stop time of punch output (see ACSTRT, above)	PL4
ACUSER	16 bytes of user information from the * $$ JOB card	CL16
ACNAME	VSE/POWER job name from the * $$ JOB card	CL8
ACNUMB	Job number assigned by VSE/POWER (same as that of associated reader queue entry for first $$ PUN statement)	BL2
ACIDEN	Record ID (P)	CL1
ACCANC	VSE/POWER cancel code: X'10' = Normal end of VSE/POWER job or task - The associated z/VSE job(s) may have been canceled by the system nevertheless. X'30' = PSTOP command was issued. The code is not stored in the account record if the EOJ option was specified in the PSTOP command. X'40' = PFLUSH command was issued. X'70' = The job was canceled due to an I/O error.	BL1
ACFLG1	Account flag byte 1: ACF1CE20 X'80' ON = ACDATE is 20yy OFF = ACDATE is 19yy	BL1
end of header		

Table 8. Punch Account Record (continued)

Field Name	Description	Field Type & Length
PUNADR	Punch device or RJE-line address (cuu), SNA, or GSP.	CL3
PUNFRM	FROM remote ID	BL1
PUNTO	TO remote ID	BL1
PUNOCL	Punched output class	CL1
PUNOPR	Punched output priority number	CL1
PUNNUM	Number of records punched (see also PUNEXR).	BL4
PUNTRK	Number of DBLK groups for output storage. The field is set to zero if the output was spooled to tape.	BL2
PUNSUF	Job suffix (segment) number assigned by VSE/POWER.	BL1
PUNCOP	Number of punched copies (if more than one, the statistics are the totals for all copies).	BL1
PUNFOR	Punch-forms identification.	CL4
PUNEXR	Number of additional cards punched due to restart, separator cards, or extra copies.	BL4
PUNNODE	Name of own node (system) in the network.	CL8
PUNTOUS	Destination-user (TO) identification.	CL8
PUNFRNO	Name of originating (FROM) node.	CL8
PUNFRUS	Originating-user (FROM) identification.	CL8
PUNOJ#	Original job number, if one exists.	BL2
PUNACCT	Network account number	CL8
	Reserved	CL2
PUNRINS	Records inserted by OUTEXIT routine	BL4
PUNRDEL	Records deleted by OUTEXIT routine	BL4
PUNPGRNM	Programmer's name	CL20
PUNBLDG#	Programmer's building number	CL8
PUNROOM#	Programmer's room number	CL8
PUNDEPT#	Programmer's department number	CL8
PUNDIST	Distribution code	CL8

Layout of the Reader Account Record

VSE/POWER builds a *reader account record* for every VSE/POWER job submitted for spooling. Whether the queue entry has actually been queued is indicated by the VSE/POWER cancel code. VSE/POWER does not build reader account records for a writer-only partition.

Table 9. Reader Account Record

Field Name	Description	Field Type & Length
ACDATE	Processing date in the format as defined to the system.	CL8
ACSTRT	Start time of read (0hhmmssf, where f = sign).	PL4
ACSTOP	Stop time of read (see ACSTRT, above).	PL4
ACUSER	16 bytes of user information from the * $$ JOB statement.	CL16
ACNAME	VSE/POWER job name from the * $$ JOB or the // JOB statement.	CL8
ACNUMB	Job number assigned by VSE/POWER.	BL2
ACIDEN	Record ID (R)	CL1
ACCANC	VSE/POWER cancel code: X'10' = Normal end of VSE/POWER job or task - the associated z/VSE job(s) may have been canceled by the system nevertheless. X'30' = PSTOP command was issued. The code is not stored in the account record if the EOJ option was specified in the PSTOP command. X'40' = PFLUSH command was issued. X'60' = The job was canceled via JOBEXIT. X'70' = The job was canceled due to an I/O error.	BL1
ACFLG1	Account flag byte 1: ACF1CE20 X'80' ON = ACDATE is 20yy OFF = ACDATE is 19yy	BL1
end of header		
RDRADD	Reader device or line address (cuu), SNA, or PSP for submission from a partition	CL3
RDRFRM	FROM remote ID	BL1
	Reserved	BL1
RDRICL	Input class	CL1
RDRIPR	Input priority number	CL1
RDRNUM	Number of records read (including record added or deleted by a reader exit routine)	BL4
RDRTRK	Number of DBLK groups for input storage	BL2
RDRNODE	Name of own node (system) in the network	CL8
RDRFRUS	Originating-user (FROM) ID	CL8
RDRACCT	Network account number	CL8

Layout of the Transmitter/Receiver Account Record

VSE/POWER builds a *transmitter/receiver-account* record for every job or output transmission via a connection or during a session.

Transmitter/Receiver Account Record

Table 10. Transmitter- or Receiver-Account Record

Field Name	Description	Field Type & Length
ACDATE	Processing date in the format as defined to the system	CL8
ACSTRT	Start time (0hhmmssf, where f = sign)	PL4
ACSTOP	Stop time (0hhmmssf, where f = sign)	PL4
ACUSER	User information	CL16
ACNAME	Job name	CL8
ACNUMB	Job number	BL2
ACIDEN	Record ID (V = receiver; M = transmitter)	CL1
ACCANC	VSE/POWER cancel code: X'10' = Normal end of VSE/POWER transmitter or receiver task X'30' = PSTOP command was issued. The code is not stored in the account record if the EOJ option was specified in the PSTOP command. X'40' = PFLUSH command was issued. X'60' = The job was flushed via NETEXIT/XMTEXIT. X'70' = The job was canceled due to an I/O error. X'80' = The job or output transmission was canceled due to a receiver-task stop or a transmitter-task stop at the other end.	BL1
ACFLG1	Account flag byte 1: ACF1CE20 X'80' ON = ACDATE is 20yy OFF = ACDATE is 19yy	BL1
end of header		
NACLAD	Line address (cuu) or 'SNA' or 'TCP' or 'SSL'	CL3
NACQTYP	Queue type (R = reader; L = list; P = punch)	CL1
	Reserved	BL1
NACCLAS	Class of job/output	CL1
NACPR	Processing priority in local queue	CL1
NACCNTD	Data record count	BL4
NACORGJ#	Original job number from job reader	BL2
NACSUF	Job suffix (segment) number	BL1
NACCOP	Number of copies	BL1
NACCNTC	Control record count	BL2
	Reserved	BL2
NACON	Name of originating node	CL8
NACOUS	Name (user ID) of remote originator	CL8
NACTN	Name of destination node	CL8
NACTUS	Destination-user ID	CL8
NACCURR	Current (own z/VSE) node name	CL8
NACADJ	Adjacent node name	CL8
NACACCT	Network account number	CL8
NACINR	Records inserted by NETEXIT/XMTEXIT routine	BL4
NACDLR	Records deleted by NETEXIT/XMTEXIT routine	BL4

Layout of the RJE,BSC Account Record

VSE/POWER builds an *RJE,BSC account record* for an RJE,BSC user session when it processes a sign-off or when a line stop occurs.

Table 11. RJE,BSC Account Record

Field Name	Description	Field Type & Length
BSCDTE	Processing date in the format as defined to the system	CL8
BSCSGN	SIGNON time (0hhmmssf, where f = sign)	PL4
BSCSGF	SIGNOFF time (0hhmmssf, where f = sign)	PL4
BSCUSE	16 bytes of user information from the SIGNON command	CL16
BSCPAS	Line password	CL8
BSCIRS	Number of invalid responses during transmission (see the Note below)	BL2
BSCIDN	Record ID (T)	CL1
BSCSFC	SIGNOFF code (any combination of the codes may occur): X'01' = Normal SIGNOFF X'02' = SIGNOFF forced due to PSTOP cuu X'04' = SIGNOFF forced due to excessive idle time X'08' = SIGNOFF forced due to unrecoverable I/O error X'10' = SIGNOFF forced due to PEND or PSTOP cuu,EOJ X'20' = SIGNOFF forced by lack of processor storage X'40' = SIGNOFF forced due to PSTOP cuu,FORCE X'80' = SIGNOFF forced due to line stop at last I/O	BL1
BSCTEC	Terminal (workstation) error count	BL1
BSCLAD	Line address	CL3
BSCRID	Remote ID	BL1
BSCFLG1	BSC account flag byte 1: BSC1CE20 X'80' ON = BSCSOD is 20yy OFF = BSCSOD is 19yy	BL1
BSCTRAN	Transmission count per session (see the Note below).	BL2
BSCTCNT	Time-out count per session (see the Note below)	BL2
BSCERR	Error count per session (see the Note below)	BL2
BSCSOD	SIGNOFF date (mmddyy)	CL6

Note: Comparing fields BSCTRAN and BSCTCNT gives an indication of idle time per session. Comparing fields BSCTRAN, BSCTCNT, and BSCERR gives an indication of line quality.

Layout of the RJE,SNA Account Record

VSE/POWER builds an *RJE,SNA account record* when an RJE,SNA user session ends.

Table 12. RJE,SNA Account Record

Field Name	Description	Field Type & Length
SNADTE	Processing date in the format as defined to the system	CL8
SNASGN	SIGNON time (0hhmmssf, where f = sign)	PL4
SNASGF	SIGNOFF time (0hhmmssf, were f = sign)	PL4
SNAUSE	16 bytes of user information from the SIGNON command.	CL16
SNALUN	Logical unit name	CL8
SNAFLG1	SNA account flag byte 1: SNA1CE20 X'80' ON = SNADTE is 20yy OFF = SNADTE is 19yy	BL1
	Reserved	CL1
SNAIDEN	SNA record ID (S)	CL1
SNATERM	Session termination code: X'01' = normal termination (LOGOFF or SIGNOFF) X'02' = abnormal termination	BL1
SNARID	Remote ID	BL4

Layout of the System-Up Account Record

VSE/POWER builds a *system-up account record* on completion of VSE/POWER startup.

Table 13. System-Up Account Record

Field Name	Description	Field Type & Length
PWRDTE	Processing date in the format as defined to the system	CL8
PWRSGN	Startup time	PL4
PWRFLG1	Startup flag byte 1: PWR1CE20 X'80' ON = PWRDTE is 20yy OFF = PWRDTE is 19yy	BL1
	Reserved	BL3
PWRVER	Version/Modification level	CL4
PWRLEV	Level ID	CL4
PWRPARSZ	Partition size	BL4
PWRGETSZ	GETVIS size	BL4
PWRRELSZ	Reserved processor (real) storage size	BL4
PWRPART	Partition ID (BG or Fn)	CL2
PWRFLAG	Feature flags	CL4
PWRIDEN	Record ID (U)	CL1
PWRDXTN	Number of data file extents	BL1
PWRDTRK	Number of tracks/blocks in the data file	BL4
PWRQTRK	Number of tracks/blocks in the queue file	BL4
PWRATRK	Number of tracks/blocks in the account file	BL4

Layout of the Spool-Access-Connect Account Record

VSE/POWER builds a *spool-access-connect account record* when an established communication path is terminated, normally or abnormally.

Table 14. Spool-Access-Connect Account Record

Field Name	Description	Field Type & Length
XCODATE	Processing date in the format defined to the system	CL8
XCOSTRT	Connection start time (0hhmmssf, where f = sign)	PL4
XCOSTOP	Connection stop time (0hhmmssf, where f = sign)	PL4
XCOAPPL	XPCC application ID	CL8
XCOMSG#	Number of messages returned in response to a CTL or PUT request	BL4
XCOCTL#	Number of CTL requests	BL4
XCOTERM	Connection-termination code:	BL1
XCOTCOK	X'01' = Normal end of communication	
XCOTCPD	X'02' = Termination because of a PEND command	
XCOTCPP	X'04' = Termination because of a PSTOP command (but not if FORCE is specified)	
XCOTCAT	X'08' = Abnormal end by user application	
XCOTCUE	X'10' = Severe error in the application program	
XCOTCKL	X'20' = Termination because of a PSTOP command (if FORCE is specified)	
XCOTCSE	X'40' = System or VSE/POWER failure	
XCOFLG1	SAS connection flag byte 1: XCO1CE20 X'80' ON = XCODATE is 20yy OFF = XCODATE is 19yy	BL1
XCODEVN	Device name (as defined to the device-owning sub-system)	CL8
XCOIDEN	Record ID (C)	CL1

Layout of the Spool-Access-Operation Account Record

VSE/POWER builds a *spool-access-operation account record* for a PUT or a GET service when the processing for a queue entry is finished. If data is added to an appendable output, VSE/POWER builds an account record of this type every time a program finishes appending data to this output.

No accounting is performed for CTL requests or for output queue entries that are held in the queue with a disposition of X (because of an abnormal termination of VSE/POWER).

CTL requests for PDISPLAY queue, however, may generate GET operation account records. For suppression of these records and for more details, refer to "Retrieving Messages" on page 67.

Spool-Access Operation Account Record

Table 15. Spool-Access-Operation Account Record

Field Name	Description	Field Type & Length
XSPDATE	Processing date in the format as defined to the system	CL8
XSPSTRT	Start time of processing (0hhmmssf, where f = sign)	PL4
XSPSTOP	Stop time of processing (0hhmmssf, where f = sign)	PL4
XSPUSER	16 bytes of user information (field SPLDUI of the PWRSPL DSECT)	CL16
XSPNAME	Name of job (or report)	CL8
XSPNUMB	Job number as assigned by VSE/POWER	BL2
XSPIDEN	Record ID (X)	CL1
XSPCANC	VSE/POWER cancel code: X'10' = Normal end of VSE/POWER job or task - the associated z/VSE job(s) may have been canceled by the system nevertheless. X'30' = A PSTOP or PEND command was issued. X'40' = A PFLUSH command was issued. X'50' = A purge request (during a queue entry re-trieval) or a PDELETE command was issued. X'90' = A quit request was issued. X'A0' = The operation was terminated because a severe error occurred or the system failed to maintain the communication path. X'B0' = A CLOSE request was issued. X'C0' = Canceled due to lack of disk space. X'D0' = A 'quit-and-lock' request was issued.	BL1
XSPFLG1	SAS operation flag byte 1: XSP1CE20 X'80' ON = XSPDATE is 20yy OFF = XSPDATE is 19yy	BL1
XSPREQT	Request type (G = GET request; P = PUT request).	CL1
XSPQUID	Queue type (R = reader; L = list; P = punch).	CL1
XSPJSUF	Job-suffix (output segment) number	BL1
XSPCLSS	Class	CL1
XSPPRIO	Priority	CL1
XSPDISP	Disposition	CL1
XSPCOPY	Number of copies (output only).	BL1
XSPCPYG	Copy groupings (3800 output only).	BL8

Table 15. Spool-Access-Operation Account Record (continued)

Field Name	Description	Field Type & Length
XSPTRK#	Number of DBLK groups occupied on disk (see Note 1, below)	BL2
XSPOJ#	Original job number	BL2
XSPREC#	Number of records. The value includes the control record, even if a spool-access user has not specified CTLREC=YES in the applicable PWRSPL macro. SPL records returned by VSE/POWER are not included in this record count. See also Note 1, below.	BL4
XSPEXR#	Number of extra records	BL4
XSPLNE#	Total number of lines or cards (output only); see also Notes 1 and 2, below.	BL4
XSPEXL#	Number of extra lines or cards because of separator pages or cards, or because of records repeated as a result of a restart (applies only to GET requests).	BL4
XSPPGE#	Total number of pages, excluding any double counting of existing pages encountered during GET/PUT restart (output only); see also Notes 1 and 3 below.	BL4
XSPEXP#	Number of extra pages such as separator pages or pages passed repeatedly as a result of a restart.	BL4
XSPFORM	Forms identification (applies to output only).	CL8
XSPFLSH	Flash identification (applies to 3800 output only).	CL4
XSPTONM	Name of destination node	CL8
XSPTOUS	Destination-user ID	CL8
XSPRQUS	Requesting-user ID	CL8
XSPRQAP	Requesting XPCC application ID	CL8
XSPNODE	Name of your own node	CL8
XSPRINS	Records inserted by OUTEXIT routine	BL4
XSPRDEL	Records deleted by OUTEXIT routine	BL4
XSPACCT	Network account number	CL8

Note:

1. The count applies to and is shown for appendable output only.
2. The line-number count is set to the record count if the queue entry's record format is SCS, 3270 data stream, BMS, CPDS, or Escape mapping.
3. The total page count is not meaningful for a list queue entry containing data in the BMS mapping or the 3270 data stream format. For output of this type, every record is considered to be a page. If the queue entry contains data in the SCS or the escape mapping format, the total page count is not meaningful either and, therefore, set to zero.

The PUTACCT Macro: Adding User Information to Account Records

You can add information to every execution account record by writing a routine under the name of $JOBACCT that makes use of the PUTACCT macro. Before you issue this macro in your routine, save registers 0 and 1, because they will be overwritten by VSE/POWER. Link this routine as phase $JOBACCT in your system's sublibrary IJSYSRS.SYSLIB and see that job accounting is specified in the VSE/POWER control table generated by the POWER® generation macro, refer to *VSE/POWER Administration and Operation*, SC34-2625. This causes the IBM-supplied dummy phase named $JOBACCT to be overwritten. For guidance on how to write the routine, refer to *z/VSE Guide to System Functions*, SC33-8312; an example for the use of the PUTACCT macro is given below.

Job Control calls your $JOBACCT program at the end of every job or job step.

VSE/POWER ignores the macro if job accounting has not been defined in the POWER generation macro.

Requirements For the Caller

AMODE:
24 or 31

RMODE:
24

ASC Mode:
Primary

Format of the Macro

```
►►──┬──────┬──PUTACCT (reg1),(reg2)──────────────────────────────────►◄
    └─name─┘
```

(reg1),(reg2)
For the operands reg1 and reg2, specify two different general registers, but not registers 0 and 1. When you issue the macro, the registers must contain the following:

reg1
The 24-bit address of the area that contains the additional information.

reg2
The length of the above mentioned area.

The maximum length of the area may not exceed 2,008 bytes minus the length of the execution account record set up by VSE/POWER as described in Table 5 on page 13.

Return Codes from the PUTACCT Macro

Successful completion of the PUTACCT macro is indicated to the issuing program by a return code of 0 in register 0. If the operation fails, register 0 contains the return code listed below.

Code	Meaning
X'04'	The VSE/POWER execution account record (with its variable length SIO

table part) extended by the user provided PUTACCT area, exceeds the maximum record length of 2008 bytes. The request is ignored.

Example of the PUTACCT Macro

The following example inserts additional information behind the VSE/POWER execution account records:

```
          ... ... ...
         COMRG REG=R4          GET PARTITION COMMUNICATION REGION
         USING CMRG,R4         DECLARE ADDRESSABILITY
         TM    POWFLG1,X'80'   ACCOUNT SUPPORT FOR THIS PARTITION
         DROP R4
         BNO   EXIT            BRANCH IF NOT
         LA    R2,ADAC         ADDRESS ADDITIONAL INFORMATION
         LA    R3,L'ADAC       LENGTH ADDITIONAL INFORMATION
         PUTACCT (R2),(R3)     PASS INFORMATION TO VSE/POWER
 EXIT    DS    0H
         BR    RE              RETURN TO $JOBCTLN
          ... ... ...
 ADAC    DC    C'ADDITIONAL ACCOUNT INFORMATION'
 R2      EQU   2               REGISTER 2
 R3      EQU   3               REGISTER 3
 R4      EQU   4               REGISTER 4
 RE      EQU   14              REGISTER 14
          ... ... ...
 CMRG    MAPCOMR
          ... ... ...
+POWFLG1 DS    *
          ... ... ...
         END
```

Chapter 3. Output Segmentation

VSE/POWER job output can be segmented, that is, part of the output from a job can be printed or punched before the entire job is finished.

Several types of segmentation are possible, depending on the event that initiates the segmentation.

- Program-driven output segmentation

  In your application program, you may use the VSE/POWER IPWSEGM macro (or SEGMENT macro as described in Appendix B, "Output Segmentation by SEGMENT Macro," on page 371) to separate the output whenever your program logic decides to do so.

  Also, the LFCB macro may be used, which causes segmentation before loading the new FCB. If your output is directed to an IBM 3800, you may use a z/VSE // SETPRT JCL statement requesting a printer setup to cause segmentation.

  These are setup requests that require operator intervention and, therefore, always cause segmentation.
- Spool-Access PUT-OUTPUT segmentation is described in "Requesting Output-Segmentation" on page 125.

The other types of output segmentation are described in *VSE/POWER Administration and Operation*, SC34-2625:

- Command-driven output segmentation
- Count-driven output segmentation
- Data-driven output segmentation
- Multivolume tape segmentation

IPWSEGM Macro - Extended Output Segmentation

The IPWSEGM macro can be used for controlling output segmentation for a job running in a VSE/POWER-controlled partition. Calling the IPWSEGM macro means that all output passed thus far to VSE/POWER for a certain printer or punch device should now be made available as a LST/PUN-queue entry - called **last** segment. More output to come is to be collected as the **next** segment, with attributes as specified by the * $$ LST/PUN statement passed along by the JECL= operand of the IPWSEGM macro. IPWSEGM is ignored for suppressed output spooling, that is, if DISP=N was specified in the * $$ LST or * $$ PUN statement within a running job.

As compared to the SEGMENT macro (see Appendix B, "Output Segmentation by SEGMENT Macro," on page 371), the IPWSEGM macro offers **extended** segmentation functions, such as:

- Passing attributes for the **next** segment in a 1024-byte area, thus providing ample space for a nearly unlimited series of contiguous operands of the * $$ LST/PUN statements.
- Acquiring default attributes for the **next** segment by specifying no macro operand at all (apart from DEVADDR=).
- Keeping all active output attributes of the **last** segment for the **next** segment (KEEP=YES operand).

- Extending and/or overwriting all active output attributes of the **last** segment for the **next** segment by specifying KEEP=YES and passing a * $$ LST/PUN statement at the same time.
- Returning 'queue-id', 'class', 'jobname', 'jobnumber', 'jobsuffix' (in case RBS= requested), and the VSE/POWER internal 'queue-entry number' of the **last** segment to the user program. These direct-access search arguments provide unique specifications for expedited spool-access support GET service (see "Direct Queue Entry GET Access to the RDR/LST/PUN/XMT Queues" on page 79) and CTL-service (see "Direct Queue Entry CTL Access" on page 68)
- Supporting re-enterable coding through the MFG= operand.
- Returning unique return and feedback codes for analysis of failure.
- Addressing VSE/POWER via a multi-threaded path, not tying up any z/VSE resource.
- Defining duplicates for the **next** segment using the operand DUP=YES in the supplied IPWSEGM's JECL. DUP=YES provides the same functionality as * $$ LSTDUP and * $$ PUNDUP supplied in a VSE/POWER job.

Note: The IPWSEGM macro call results in a spooled I/O request for the specified device (DEVADDR=), and macro completion may in extreme cases depend on storage or spool-space shortage of the VSE/POWER partition.

When calling macro IPWSEGM, the application program
- should consider saving register 0, 1, and 15, which are used by VSE/POWER,
- should include the new mapping macro IPW$MXD already **before** the CSECT that contains the segment request. Macro IPW$MXD
  - is required during macro assembly time, and
  - may be used to reference return information
- should avoid using symbols and labels starting with **'$'**, which is reserved for IPWSEGM and IPW$MXD.

Generation of Duplicate LST and PUN Output

To generate LST or PUN output with one or more duplicates using IPWSEGM for program-driven segmentation, do the following:

1. Specify the duplicates of the first output segment in the JECL statements of a job: in * $$ LST and * $$ LSTDUP statements for LST output or in * $$ PUN and * $$ PUNDUP statements for PUN output.
2. Specify duplicates of the next segment in the JECL statement * $$ LST or * $$ PUN, which is supplied by the IPWSEGM macro call.

   For each duplicate, include DUP=YES followed by at least one of the allowed duplicate operands for * $$ LSTDUP or * $$ PUNDUP in the JECL statement. The duplicate operands are used to define output properties different to the properties of the master output. The duplicate operands are listed below:
   - JNM=jobname
   - CLASS=class
   - DISP=disposition
   - PRI=priority
   - COPY=number_of_copies
   - DEST=mode_id|(node_id,user_id)|
   - TDISP=disposition
   - REMOTE=remote_id

- DIST=distcode|NULL
- SYSID=n|N
- UINF=user_info
- EXPDAYS=nnn
- EXPHRS=hh
- EXPMOM=NULL

When specifying KEEP=NO in IPWSEGM, all duplicate definitions for the previous segment will be dropped. Specifying KEEP=YES will keep duplicate definitions and can either be replaced by defining new duplicates or explicitly cleared by the operand DUP=NO, which must be the last operand in the supplied * $$ LST or * $$ /PUN statement. The next table shows how IPWSEGM updates the duplicate definition for the next segment.

	KEEP=YES	**KEEP=NO**
DUP=YES[1]	New duplicate(s)	New duplicate(s)
DUP=NO	No duplicate	Error $MX0CDNI
No DUP specified	As defined	No duplicate
[1] DUP=YES with one or more duplicate operands (repeated for additional duplicates)		

The following example illustrates how to specify the JECL for duplicate LST output and the subsequent results.

```
Job Stream:                                  Continuation Column 72-----+
------------                                                            |
* $$ JOB ...                                                            V
* $$ LST JNM=LMAST1ST,CLASS=B,DISP=K,LST=FEE,                           C
* $$ LSTDUP JNM=LDUPA1ST,                                               C
* $$ CLASS=C,                                                           C
* $$ LSTDUP JNM=LDUPB1ST,TDISP=L,DEST=OTHERNOD
// MY JOB
// EXEC MYAPPL
/&
* $$ EOJ ...
```

Program MYAPPL issues IPWSEGM macro call with KEEP=NO and the following * $$ LST statement:

```
'* $$ LST JNM=LMAST2ND,CLASS=B,DISP=K,LST=FEE,DUP=YES,JNM=LDUPA2ND,
CLASS=C,DUP=YES,JNM=LDUPB2ND,TDISP=L,DEST=OTHERNOD,DUP=YES,JNM=LDUPC'
```

When this job is started, the output spooled for device FEE creates queue entry LMAST1ST. When IPWSEGM segments the output, LMAST1ST and its duplicate LDUPA1ST are added to the LST queue and the duplicate LDUPB1ST is added to XMT queue. The next segment, LMAST2ND, is started with three duplicates LDUPA2ND, LDUPB2ND, and LDUPC. When the job ends, the last segment is created together with its duplicates. For additional explanations and details about duplicates, refer to the section "Duplication of Output Spool Entries" in *VSE/POWER Administration and Operation*, SC34-2625.

Requirements for the Caller

AMODE:
24 or 31

RMODE:
24 (or ANY: see "Residency Mode Considerations" on page 38)

ASC Mode:
Primary

Format of the Macro

```
                                                  ┌─,KEEP=NO──┐
►►──┬──────┬──IPWSEGM DEVADDR=──┬─SYSxxx──┬──┼───────────┼──────────────────────────►
    └─name─┘                    └─(reg1)──┘  └─,KEEP=YES─┘

►──┬────────────────────────────────────────────────────┬──┬──────────────────────────┬──►◄
   │                          ┌─,JECLN=71──────────┐     │  │                  (1)      │
   └─,JECL=──┬─addr───┬──┼────────────────────┼─────┘  └─,MFG=──┬─area───┬──────────┘
             └─(reg2)─┘  └─,JECLN=──┬─len────┬─┘                  └─(reg4)─┘
                                    └─(reg3)─┘
```

Notes:

1 The MFG operand is required if the IPWSEGM macro expansion is to be re-enterable.

DEVADDR=SYSxxx | (reg1)
For SYS*xxx*, specify the system or programmer logical unit assigned to the device on which segmentation is to occur. For *reg1*, if you chose register notation, specify any register (apart from register 0, 1, and 15) that points to a six-byte field containing a SYS*xxx* constant.

If you supplied a spooled device specification by the

- LST operand of your * $$ LST JECL statement, or
- PUN operand of your * $$ PUN JECL statement,

the LST/PUN operand values are ignored.

KEEP=YES | NO
This operand specifies to keep all output attributes of the **last** segment and let them become effective for the **next** segment. For combining the KEEP= and JECL= operands, refer to the description of the JECL= operand.

JECL=addr | (reg2)
This operand points to an area that contains an * $$ LST or * $$ PUN statement whose operands provide attributes for the **next** segment.

For *addr*, specify the area's label in your program.

For *reg2*, if you chose register notation, specify any register (apart from register 0, 1, and 15) that contains the address of the JECL area.

Whatever notation you use, make sure that the JECL area lies below the 16MB line and that it resides either in your partition or in the dynamic space GETVIS area.

The maximum area length is 1024 bytes and it must contain a JECL statement with a series of operands delimited by comma, followed by a blank and, optionally, by a comment. The statement may be:

1. * $$ LST/PUN 'without' operands, to obtain default attributes for the **next** segment, or
2. * $$ LST/PUN 'with' operands, to obtain the specified attributes for the **next** segment.
3. * $$ LST/PUN 'with' or 'without' operands and 'with' operand DUP=YES. The operand DUP=YES must be followed by at least one operand permitted for * $$ LSTDUP/PUNDUP. These operands will override the duplicate attributes (if any) specified for the **last** segment and remain in effect for the **next** segment. To request more than one duplicate, an additional operand DUP=YES (with permitted operands from the subset for *$$ LSTDUP/PUNDUP) must be specified for each duplicate. Note that IPWSEGM KEEP=YES will keep previously defined duplicates unless a new set of duplicates is specified by (multiple) DUP=YES or until IPWSEGM KEEP=YES with DUP=NO as last operand of JECL statement resets specification of duplicates.

Note:

1. If JECL statements are passed to VSE/POWER through the IPWSEGM interface, they must adhere to the following conventions:
   a. They must be coded as for example '* $$ LST ',
      1) starting in byte 1 of the JECL area,
      2) with exactly one blank delimiter between *, $$, and LST,
      3) with at least one blank (but not more than 247 blanks) before optional operands occur,
      4) and with at least one trailing blank after the last valid operand, provided the JECLN operand specifies a length that extends beyond the last character of the operand.
   b. They must have a minimum length of 9.
2. Through the IPWSEGM interface, VSE/POWER converts any passed JECL statements to uppercase, but
   a. Does not accept JECL statements in positional format,
   b. Does not offer incorrectly specified JECL for correction on the console-only for information by message 1Q5JI and 1Q5GI.

   Instead, the macro call fails with a unique return code.
3. When the JECL= operand is omitted, the **next** segment will receive
   a. default attributes, if KEEP=NO is specified (default) or
   b. all attributes of the **last** segment, if KEEP=YES is specified

   When the JECL= operand is specified, the **next** segment will receive
   a. The specified attributes if KEEP=NO is specified (default), or
   b. All attributes of the last segment altered by the specified operands, if KEEP=YES is specified. When an operand of a JECL statement occurs more than once, the last specification becomes effective.
4. The JECL statement continuation rules, as described in *VSE/POWER Administration and Operation*, SC34-2625, do **not** apply; instead, the desired operands may be placed beyond column 71.

5. To generate duplicates for the next segment, specify first all attributes for the original output and then for each duplicate add DUP=YES followed by the attributes for this duplicate.

   For example, the JECL statement * $$ LST JNM=LST1,CLASS=8,DISP=D,

   DUP=YES,JNM=OUT2,DISP=L,DUP=YES,JNM=LST3,CLASS=T

   will produce LST output LST1 in class 8 with disposition D, the first duplicate OUT2 in class T with disposition L, and the second duplicate LST3 in class T with disposition D.

JECLN=len | 71 | (reg3)

This operand provides the length of the JECL area. This area must be at least 9 bytes, up to a maximum of 1024 bytes.

For *len*, specify the length as a self-defining term.

For *reg3*, if you chose register notation, specify any register (apart from register 0, 1, and 15) that contains the length of the JECL area.

MFG=area | (reg4)

This Macro Format Generation (MFG) operand is only required if the IPWSEGM macro expansion is to be re-enterable, that is, if the generated macro parameter area should be reusable by parallel requests.

For *area*, specify the label in your program of the parameter area.

For *reg4*, if you chose register notation, specify any register (apart from register 0 or 15) that points to the parameter area. It is recommended to use register 1, because that is used by the macro for addressing anyhow.

Whatever notation you use, make sure that the MFG area lies below the 16MB line and that it resides either in your partition or in the dynamic space GETVIS area.

VSE/POWER adjusts the used part of the MFG area to a double-word boundary. Therefore, after IPWSEGM completion, you should use the VSE/POWER returned register 1 value to address the parameter area correctly.

The provided parameter area must have a length as defined by $MXLEN of the mapping macro IPW$MXD, which describes the area.

When the MFG operand is not used, IPWSEGM will establish a parameter area within the macro expansion.

Return Codes from the IPWSEGM Macro

The IPWSEGM macro provides completion return codes for a

- Static macro expansion at program assembly time by failure MNOTEs with respect to correctness of specified macro operands.
- Dynamic macro request at program execution time by return code (RC) and SeGment FeedBack code (SGFB) in the two low order bytes of register R15:

```
1. RC/SGFB = X'0000' for a successful IPWSEGM request
2. RC/SGFB = X'00mm' for a successful IPWSEGM request
                     with unexpected conditions (warning)

3. RC/SGFB = X'0404' for a failure because VSE/POWER is not
                     active
4. RC/SGFB = X'08nn' for failures detected by the
                     Segment Interface Routine
5. RC/SGFB = X'0Cpp' for failures detected by the VSE/POWER
                     execution processor modules.
   Note: For better tracking of segmentation request failures
   and for better synchronization of job execution with
   console message flow, all X'0Cpp' request failures are
```

```
also recorded on the central operator console by the
two informational messages:
1)1Q5JI   presenting the incorrect LST/PUN statement
2)1Q5GI   naming the failing jobname, etc.
```

For details refer to the $MXRRC/$MXRFB mnemonics provided by the IPW$MXD macro.

At request completion:

- R15 contains a unique return and feedback code that indicates incorrect usage of macro operands, unsuccessful, or successful processing.
- R1 addresses the macro parameter area, which may be interpreted using the mapping macro IPW$MXD. The returned attributes of the last created segment can be found at label $MXSQI, where the following is presented:
  - Queue-id (1 byte - R/L/P/X),
  - Class (1 byte),
  - Output suffix (1 byte) including 'last segment indicator'. Only used if the last segment was controlled by RBS= segmentation,
  - Job name (8 bytes),
  - Job number (4 bytes),
  - Unique queue-entry number (4 bytes) of the created segment. When spooling to tape using DISP=T, the queue entry number is hex zero.

  Additional returned attributes of the last created segment can be found at label $MX2DP, where the following is presented:
  - Disposition (1 byte)
  - Transmission disposition (1 byte)
  - Priority (1 byte)
  - Number of copies (1 byte, binary)
  - Target SYSID (1 byte, X'00' if empty)
  - Forms-id (4 bytes, X'40' if empty)
  - FCB name (8 bytes, X'00' if empty)
  - UCS name (8 bytes, X'00' if empty)
  - Target node (8 bytes, X'40' if empty)
  - Target user (8 bytes, X'40' if empty)
  - VM Distribution code (8 bytes, X'40' if empty)
  - User Information (16 bytes, X'40' if empty)

  If not empty, then 4/8/16 byte character fields are presented leftbound, padded to the right with blanks.

  The 'returned attributes' fields contain **valid** information if the R15 RC/SGFB code is X'0000', X'0010', or X'0014'.

  And these fields are hex zero, apart from 'job name', if the R15 RC/SGFB code is:
  1. X'0004', when the collected output contained no user data (and RBS segmentation was not active currently)
  2. X'000C', when the collected output was purged due to the PURGE operand of the * $$ LST/PUN statement.

  But these fields are all hex zero if the R15 RC/SGFB code is:
  1. X'0404', when VSE/POWER was not active
  2. X'08nn', when the IPWSEGM Interface routine of VSE/POWER detected inconsistencies

3. X'0Cpp', when the VSE/POWER execution processor detected inconsistencies.

Residency Mode Considerations

Because the IPWSEGM macro expansion provides I/O Command Control Blocks (CCB and CCW) for communication with VSE/POWER, the macro must not be called from a program residing above the 16 MB line; in other words, the macro should be called only from a program with RMODE 24.

However, the program can circumvent RMODE limitations when, for the IPWSEGM request, dynamic areas below the 16 MB line are acquired whose addresses are specified by MFG=(regx) and (optionally) by JECL=(regy). The following steps should be observed:

1. Reserve a dynamic storage area below the 16 MB line (using GETVIS LOC=BELOW) with a length of $MXLEN (see "IPW$MXD Mapping Macro") to provide for a dynamic macro expansion area; assume addressability by regx.
2. Call IPWSEGM from above the 16 MB line as follows:

```
IPWSEGM DEVADDR=SYSxxx,MFG=(regx)
```

If you provide an explicit * $$ LST statement for the IPWSEGM request, take the following steps:

1. Reserve a dynamic storage area below the 16 MB line with a length of $MXLEN (area later addressed by regx) and another area with a length of your * $$ LST statement (area later addressed by regy).
2. Set up the * $$ LST statement in the regy area.
3. Provide the length of the * $$ LST statement for the JECLN= operand.
4. Call IPWSEGM from above the 16 MB line as follows:

```
IPWSEGM DEVADDR=SYSxxx,MFG=(regx),JECL=(regy),JECLN=...
```

IPW$MXD Mapping Macro

The IPW$MXD (Macro Extension Definition) macro generates a DSECT which describes the parameter area used by the IPWSEGM macro.

The macro has no operands.

Format of the Macro

```
►►──IPW$MXD──────────────────────────────────────────────────────────►◄
```

Table 16. IPWSEGM Parameter Area Produced by IPW$MXD Macro

Bytes Hex	Field Label	Description/Function
General IPWSEGM Request Area Part 1		
00	$MXDS	START OF PARAMETER LIST
00-0F	$MXCCB	COMMAND CONTROL BLOCK
10-17	$MXCCW	CHANNEL COMMAND WORD

Table 16. IPWSEGM Parameter Area Produced by IPW$MXD Macro (continued)

Bytes Hex	Field Label	Description/Function
18-4B	$MXRSV	SAVE AREA REGISTER 2 - 14
4C-54	$MXDJA	DEFAULT JECL STATEMENT AREA
55-57		UNUSED
Input Area to VSE/POWER		
58-5B	$MXVRS	VERSION OF PARAMETER AREA
5C-5F	$MXUNA	LOGICAL UNIT ADDRESS
5C	$MXCLS	LOGICAL UNIT CLASS
5D	$MXNUM	LOGICAL UNIT NUMBER
5E-5F		LOG. UNIT ADDRESS BYTE 2+3
60-63	$MXJCL	ADDRESS OF JECL STATEMENT
64-67	$MXJCN	LENGTH OF JECL STATEMENT
68	$MXOP1	INPUT OPTION BYTE 1
	$MX1UA	X'80' - LOG. UNIT BY ADDRESS
	$MX1PJ	X'40' - PASSED JECL OF USER
	$MX1KP	X'20' - KEEP OPTION SPECIFIED
69-6F	$MXRDI	RESERVED INPUT AREA
Description Area of Last Segment Part 1		
70	$MXSQI	QUEUE-ID OF CREATED SEGMENT (R|L|P|X)
71	$MXSCL	JOB CLASS OF CREATED SEGMENT (0-9,A-Z)
72	$MXJSF	OUTPUT SUFFIX, IF 'RBS=' USED
	$MXJSFL	X'80' - 'LAST RBS SEGMENT' FLAG
		X'7F' - RBS SEGMENT SUFFIX NUMBER (BIN)

Table 16. IPWSEGM Parameter Area Produced by IPW$MXD Macro (continued)

Bytes Hex	Field Label	Description/Function
73		UNUSED
74-7B	$MXJNM	JOB NAME OF CREATED SEGMENT
7C-7F	$MXJNB	JOB NUMBER OF CREATED SEGMENT (BINARY)
80-83	$MXQNB	BIN. Q-ENTRY NUMBER OF CREATED SEGMENT
REGISTER 15 'RC = RETURN CODE'		
84	$MXRRC	RETURN CODE WITH FOLLOWING CATEGORIES
	$MXR00	X'00' - OK, NO ERROR (PERHAPS WARNING)
	$MXR04	X'04' - INITIALIZATION ERROR
	$MXR08	X'08' - SPECIFICATION INCONSISTENCIES
	$MXR0C	X'0C' - EXECUTION PROCESSING ERROR
REGISTER 15 'SGFB = SEGMENT FEEDBACK CODE'		
85	$MXRFB	FEEDBACK CODE, USING MNEMONICS THAT NAME THE 'RC' WITH WHICH THE FEEDBACK CODE IS DELIVERED
	$MX00OK	X'00' - OK
	$MX00IG	X'04' - NOTHING SPOOLED
		X'08' - UNUSED
	$MX00PU	X'0C' - OUTPUT PURGED
	$MX00NK	X'10' - DISP=N OK, SPOOLING STOPS
	$MX00NE	X'14' - DISP=N ERROR, SET DISP=D
	$MX04PNA	X'04' - VSE/POWER NOT ACTIVE

Table 16. IPWSEGM Parameter Area Produced by IPW$MXD Macro (continued)

Bytes Hex	Field Label	Description/Function
	$MX08NPC	X'04' - PARTITION NOT POWER CONTROLLED
	$MX08NSY	X'08' - DEVADDR NOT STARTING 'SYS...'
	$MX08ILU	X'0C' - INCORRECT LOGICAL UNIT 'SYSXXX',
		NEITHER 'XXX' = 000-255 NOR 'XXX' = PCH|LST
	$MX08IPD	X'10' - INVALID PUB DEVICE FOR 'SYSXXX',
		NEITHER PRINTER NOR PUNCH TYPE
	$MX08NPS	X'14' - 'SYSXXX' NO VSE/POWER SPOOLED DEVICE
	$MX08UNA	X'18' - 'SYSXXX' UNASSIGNED OR IGNORE
	$MX08IVR	X'1C' - 'SYSXXX' INTERNAL ERROR - CALL IBM
	$MX08CDN	X'20' - 'SYSXXX' CURRENTLY DISP=N SPOOL
	$MX08PWW	X'24' - PARTITION IN 'WAITING FOR WORK',
		WITH NO VSE/POWER JOB ACTIVE
	$MX08IJL	X'28' - INCORRECT JECL LENGTH,
		JECLN NOT WITHIN LIMITS 9 - 1024
	$MX08IJS	X'2C' - INCORRECT JECL STATEMENT, NOT
		STARTING '* $$ LST ' OR '* $$ PUN '
	$MX08NMD	X'30' - NO MATCHING DEVICE TYPE OF
		'SYSXXX' VERSUS '* $$ LST/PUN'
	$MX08FCD	X'34' - CDLOAD 3800-IJDANCHX FAILS
		DUE TO RESOURCE SHORTAGE
	$MX08PNF	X'38' - CDLOAD 3800-IJDANCHX FAILS

Table 16. IPWSEGM Parameter Area Produced by IPW$MXD Macro (continued)

Bytes Hex	Field Label	Description/Function
		DUE TO PHASE NOT FOUND
	$MX08UGF	X'3C' - 'GETFLD' UNEXPECTED RETURN CODE
	$MX08UCD	X'40' - 'CDLOAD' UNEXPECTED RETURN CODE
	$MX08CSP	X'44' - CONTRADICTION 'GETFLD' VERSUS
		DEVICE ENTRY SCAN, CALL IBM
	$MX0CNOM	X'04' - NO MATCHING SPOOL DEVICE
	$MX0CDEL	X'08' - INVALID OPERAND DELIMITER
	$MX0CUNK	X'0C' - UNKNOWN KEYWORD
	$MX0CINV	X'10' - INVALID OPERAND VALUE
	$MX0CSTP	X'14' - OPERATOR CANCELLED TAPE
	$MX0CINE	X'18' - INTERNAL POWER ERROR
	$MX0CINA	X'1C' - INVALID 'JECL' ADDRESS
	$MX0CFCB	X'50' - FCB ERROR
	$MX0CDUT	X'54' - DUP=YES not supported for DISP=T
	$MX0CDUC	X'58' - DUP=YES not supported for RBC active
	$MX0CDUS	X'5C' - DUP=YES not supported for RBS active
	$MX0CDUM	X'60' - DUP=YES not supported for MT partition
	$MX0CDUW	X'64' - DUP=YES not supported for writer-only partition
	$MX0CDUI	X'68' - DUP=YES not supported for PUN with DISP=I
	$MX0CDUN	X'6C' - DUP=YES not supported for DISP=N

Table 16. IPWSEGM Parameter Area Produced by IPW$MXD Macro (continued)

Bytes Hex	Field Label	Description/Function
	$MX0CD99	X'70' - Number of duplicates exceeds 99
	$MX0CDIK	X'74' - DUP=YES and no next valid operand
	$MX0CDID	X'78' - DUP=YES and invalid DISP=T|N|I
	$MX0CDNI	X'7C' - DUP=NO invalid for KEEP=NO or not last or specified after DUP=YES
86-8B	$MXRDR	RESERVED RETURN AREA
General IPWSEGM Request Area Part 2		
Description Area of Last Segment Part 2		
8C	$MX2DP	DISPOSITION OF CREATED SEGMENT
8D	$MX2TDP	TRANSM. DISPOSITION OF CREATED SEGMENT
8E	$MX2PY	PRIORITY OF CREATED SEGMENT
8F	$MX2NC	NUMBER OF COPIES OF CREATED SEGMENT
90	$MX2SID	TARGET SYSID OF CREATED SEGMENT
91-94	$MX2FI	FORMS-ID OF CREATED SEGMENT
95-98	$MX2FI2	FORMS-ID EXTENSION (RESERVED)
99-A0	$MX2FCB	FCB OF CREATED SEGMENT
A1-A8	$MX2UCS	UCS OF CREATED SEGMENT
A9-B0	$MX2TN	TARGET NODE OF CREATED SEGMENT
B1-B8	$MX2TU	TARGET USER OF CREATED SEGMENT
B9-C0	$MX2DIS	VM DISTRIBUTION CODE OF CREATED SEGMNT
C1-D0	$MX2UI	USER INFORMATION OF CREATED SEGMNT
D1-D4	$MX2TKN	TKN value (Binary)
D5	$MX2DUP	Number of duplicates (Binary)

Table 16. IPWSEGM Parameter Area Produced by IPW$MXD Macro (continued)

Bytes Hex	Field Label	Description/Function
D6-DB	$MXRDR2	RESERVED ATTRIBUTES FIELD
DC-E3	$MXALN	DOUBLE WORD ALIGNMENT BUFFER
	$MXLEN	X'E4' - LENGTH OF PARAMETER AREA

Note:

1. When output segmentation is requested by the IPWSEGM macro, all the already collected output by VSE/POWER for the specified device is added as an entry to the corresponding VSE/POWER queue - provided that any output had been produced by the VSE/POWER job before. If not, you are warned by the IPWSEGM register 15 RC/SGFB code $MXR00/$MX00IG=X'0004'.
2. CICS® environment considerations: The output which has been created between two segment macros in a job transaction for the specified logical unit is added to a VSE/POWER queue at the second macro request, that means before the program reaches end-of-job. For long running programs like CICS, you can use the IPWSEGM macro in a transaction to close spooling of output whenever desired. But, the specified output logical unit is unique in a CICS partition and, therefore, you may get mixed output if the same transaction runs twice at the same time, unless you have established private resource locking.
3. COBOL/VSE programs (and most likely all other LE/VSE languages) spool "double buffered" for unit record output, e.g. SYSLST. This causes problems if the VSE/POWER IPWSEGM macro is used. The last line of the current segment may appear as the first line of the next segment instead. The two I/O buffers are handled by LIOCS (Logical Input/Output Control System) and are not synchronized with the IPWSEGM macro call which expands into a SVC 0 and uses PIOCS (Physical Input/Output Control System).

   The solution to this problem is to select only one I/O buffer in the file definition of the calling high level language program in order to spool the data "single buffered".

Examples of the IPWSEGM Macro

Example 1: The following example shows how to code the IPWSEGM macro and its referenced data areas.

```
* -------------------------------------------------------------------
* INCLUDE MAPPING MACRO IPW$MXD BEFORE YOUR PROGRAM'S FIRST CSECT
* -------------------------------------------------------------------
         IPW$MXD
*
OWN      CSECT
          ... ... ...
* -------------------------------------------------------------------
* REQUEST SEGMENTATION FOR SYSLST OUTPUT DATA
*   - PROVIDE OUTPUT ATTRIBUTES FOR THE NEXT SEGMENT
*     AS DEFINED IN 'LSTCARD' JECL STATEMENT
*   - REMEMBER THAT REG. 0,1,15 ARE DESTROYED BY THE IPWSEGM CALL
* -------------------------------------------------------------------
         LA    2,LSTCARD
*
         IPWSEGM DEVADDR=SYSLST,JECL=(2)
*
```

```
* -----------------------------------------------------------------
* IF YOU WANT TO CARE ABOUT IPWSEGM REG.15 RETURN/FEEDBACK CODES,
* USE THE DETAILED CHECKING HINTS OFFERED IN EXAMPLE 2 !!!
* -----------------------------------------------------------------
          ... ... ...
       PRODUCE MORE SYSLST OUTPUT
          ... ... ...
* -----------------------------------------------------------------
* SECOND SEGMENTATION REQUEST FOR SYSLST OUTPUT DATA
*   - PROVIDE OUTPUT ATTRIBUTES FOR THE NEXT SEGMENT
*     AS DEFINED IN 'LSTCARD2' JECL STATEMENT
*   - USE 'KEEP=YES' OPTION TO KEEP NAME 'TESTOUT' FOR NEXT SEGMENT,
*     TO OVERWRITE KEPT 'FNO' & 'DISP' OPERANDS BY 'LSTOVER2' VALUES,
*     AND TO ADD EXTRA OPERANDS FROM VALUES STARTING AT 'LSTADD2'.
* -----------------------------------------------------------------
         LA    3,CARD2LEN
*
         IPWSEGM DEVADDR=SYSLST,KEEP=YES,JECL=LSTCARD2,                C
               JECLN=(3)
*
* -----------------------------------------------------------------
* IF YOU WANT TO CARE ABOUT IPWSEGM REG.15 RETURN/FEEDBACK CODES,
* USE THE DETAILED CHECKING HINTS OFFERED IN EXAMPLE 2 !!!
* -----------------------------------------------------------------
          ... ... ...
       MORE LOGIC OF YOUR PROGRAM
          ... ... ...
          ... ... ...
* -----------------------------------------------------------------
* DEFINE 'LSTCARD' JECL STATEMENT INCLUDING A TRAILING BLANK
* -----------------------------------------------------------------
LSTCARD  DC    CL71'* $$ LST JNM=TESTOUT,FNO=ACB1,DISP=H'
          ... ... ...
* -----------------------------------------------------------------
* DEFINE 'LSTCARD2' JECL STATEMENT WITHOUT A TRAILING BLANK,
* BECAUSE 'JECLN=' CONTAINS THE EXACT LENGTH OF THE STATEMENT
* -----------------------------------------------------------------
LSTCARD2 DC    C'* $$ LST '
LSTOVER2 DC    C'FNO=ACB2,DISP=L'
LSTADD2  DC    C',CLASS=B,PRI=8'
         DC    C',USER=MY-PRIVATE-INFO'
         DC    C',DEST=(ANYNODE,ANYUSER)'
CARD2LEN EQU   *-LSTCARD2
```

Example 2:
This sample job creates another VSE/POWER job in the RDR queue with new name and new input class using the DISP=I facility. To accomplish this, two IPWSEGM macro requests are required.

The first IPWSEGM macro passes a '* $$ PUN' JECL statement to VSE/POWER. This statement contains 'DISP=I' to indicate that the punch output being created from now on should be added to the RDR queue. It also contains 'CLASS=7' and 'JNM=NEWJOB2' to specify the execution class of the new job segment with the unique name 'NEWJOB2'. The next step is to punch the job control and user data to the punch device, using physical I/O control (PIOCS). A second IPWSEGM macro is required to have the collected punch output segment added to the RDR queue.

```
// JOB IPWSEGM
// OPTION CATAL
// LIBDEF *,SEARCH=PRD1.MACLIB
// LIBDEF PHASE,CATALOG=IJSYSRS.SYSLIB
 PHASE SEGMTEST,*
// EXEC ASSEMBLY,SIZE=100K
* -----------------------------------------------------------------
```

```
* INCLUDE MAPPING MACRO IPW$MXD BEFORE YOUR PROGRAM'S FIRST CSECT
* ------------------------------------------------------------------
         IPW$MXD
*
         CSECT
         PRINT GEN
         BALR  10,0
         USING *,10
* ------------------------------------------------------------------
* POINT TO THE * $$ PUN STATEMENT WITH DISP=I, THE NEW JOB CLASS AND
* THE NEW JOB NAME
* ------------------------------------------------------------------
         LA    2,PUNCARD
* ------------------------------------------------------------------
* NOW PASS FIRST IPWSEGM REQUEST TO VSE/POWER          >>> SEE NOTE 1 -->
* (REMEMBER THAT REG. 0,1,15 ARE DESTROYED BY THE IPWSEGM CALL)
* ------------------------------------------------------------------
*
         IPWSEGM DEVADDR=SYS008,JECL=(2)
*
         LTR   15,15                   SEGMENT CREATED SUCCESSFULLY,
*                                      - WITH RC/SGFB=X'0000'?
         BZ    SGM1OK1                 ..YES, GO IDENTIFY SEGMENT
* ------------------------------------------------------------------
* NECESSARY CODING TO HANDLE WARNINGS GIVEN BY RC/SGFB=X'00mm'
*     EITHER
*  1) ACCEPT ALL X'00mm' AS 'SUCCESSFUL' (shown by actual code flow)
*     OR
*  2) LOOK FOR A SINGLE RC/SGFB CODE COMBINATION   (shown by '*>')
*     OR
*  3) CHECK ALL POSSIBLE RC/SGFB CODE COMBINATIONS (shown by '*|')
* ------------------------------------------------------------------
         LR    0,15                    SAVE RC/SGFB IN REG. 0
         SRL   15,8                    LET SGFB DROP OUT RIGHT SIDE
         LTR   15,15                   IS RC=0, MACRO CALL CORRECT?
         BNZ   SGM1RCXX                .., NO GO FOR FAILURE CHECK
         B     SGM1OK2                 GO AND CONTINUE W/O SEGMENT,
*                                      APART FROM '0010'/'0014'
*>------------------------------------------------------------------
*> 2) DEMONSTRATE, HOW TO PICK OUT A SPECIFIC RC/SGFB:
*> FIND AND ACCEPT RC/SGFB=X'0004', FLAG OTHER RC/SGFB AS INVALID
*> (EXPECT ORIGINAL RC/SGFB STILL IN REG. 15)
*>------------------------------------------------------------------
*>       LR    0,15                    SAVE RC/SGFB IN REG. 0
*>       CLM   0,3,RCFB0004            NO DATA SPOOLED UP TO NOW?
*>       BE    SGM1OK2                 ..YES, CONTINUE W/O SEGMENT
*>       B     SGMERROR                GO & ISSUE ERROR MESSAGE
*>
*|------------------------------------------------------------------
*| 3a) DEMONSTRATE, HOW TO TAKE ACTION FOR ALL 'mm'
*| SGFB CODES OF RC/SGFB=X'00mm' USING A BRANCH TABLE
*| (EXPECT ORIGINAL RC/SGFB STILL IN REG. 15)
*|------------------------------------------------------------------
*|       LR    0,15                    SAVE RC/SGFB IN REG. 0
*|       SRL   15,8                    LET SGFB DROP OUT RIGHT SIDE
*|       LTR   15,15                   IS RC=0, MACRO CALL CORRECT?
*|       BNZ   SGM1RCXX                .., NO GO FOR FAILURE CHECK
*|       LR    15,0                    COPY ORIGINAL RC/SGFB
*|       SLL   15,24                   LET RC DROP OUT LEFT SIDE
*|       SRL   15,24                   OBTAIN ONLY SGFB RIGHT SIDE
** PROTECT AGAINST UNEXPECTED HIGH SGFB-CODE EXCEEDING TABLE ENTRIES
*|       CLM   15,1,MAX18              SGFB < UNEXPECTED VALUES?
*|       BL    CONT1                   ..YES, CONTINUE
*|       IC    15,MAX18                ELSE FORCE TO LAST ENTRY
** USE '04','08',...,'14' STEPS OF SGFB TO REACH BRANCH ENTRIES
*|CONT1  DS    0H
*|       B     *(15)                   BRANCH ACCORDING TO SGFB
```

```
*|       B     RF0004                   .. HANDLE RC/SGFB X'0004'
*|       B     SGMERROR                 .. HANDLE FEEDBACK X'0008',
*|*                                        WHICH IS 'UNUSED'
*|       B     RF000C                   .. HANDLE RC/SGFB X'000C'
*|       B     RF0010                   .. HANDLE RC/SGFB X'0010'
*|       B     RF0014                   .. HANDLE RC/SGFB X'0014'
*|       B     SGMERROR                 .. HANDLE TABLE END X'0018'
*|
*|RF0004 DS    0H
*|       ....  YOUR CODE TO REACT UPON $MX00IG
*|RF000C DS    0H
*|       ....  YOUR CODE TO REACT UPON $MX00PU
*|RF0010 DS    0H
*|       ....  YOUR CODE TO REACT UPON $MX00NK
*|RF0014 DS    0H
*|       ....  YOUR CODE TO REACT UPON $MX00NE
*|
* -------------------------------------------------------------------
* 3b) CODING TO HANDLE RC=X'04'...X'0C' (HIGHER RC WILL NEVER OCCUR!)
*     EITHER
*  A) FLAG ALL CASES 'INVALID', GO AND ISSUE MSG  (actual code flow)
*     OR
*  B) CHECK ALL REMAINING RC COMBINATIONS         (shown by '*%')
* -------------------------------------------------------------------
SGM1RCXX DS    0H
         B     SGMERROR                 GO & ISSUE ERROR MESSAGE
*
*%-------------------------------------------------------------------
*% B) TAKE ACTION FOR '04'...'0C' RETURN CODES USING A BRANCH TABLE
*%    (EXPECT RC IN RIGHTMOST BYTE OF REG. 15)
*%-------------------------------------------------------------------
*%       B     *(15)                    BRANCH ACCORDING TO RC
*%       B     R0004                    .. HANDLE RC/SGFB X'0404'
*%       B     R0008                    .. HANDLE RC/SGFB X'08nn'
*%       B     R000C                    .. HANDLE RC/SGFB X'0Cpp'
*%
*%R0004  DS    0H
*%       B     SGMERROR                 $MX04PNA SHOULD NEVER OCCUR
*%*                                     IN YOUR ENVIRONMENT
*%R0008  DS    0H
*%       B     SGMERROR                 GO & IDENTIFY RC/SGFB FROM
*%*                                     MSG AND CORRECT
*%*                                     - YOUR PROGRAM SPECIFICATIONS
*%*                                     - THE ENVIRONMENTAL CONDITION
*%
*%       NOTE: AGAIN YOU MAY CODE ANOTHER BRANCH TABLE TO TAKE ACTION
*%             ACCORDING TO THE SGFB CODES X'08nn'
*%
*%R000C  DS    0H
*%       B     SGMERROR                 GO & IDENTIFY RC/SGFB FROM
*%*                                     MSG AND CORRECT YOUR JECL
*%*                                     STMT PASSED TO IPWSEGM
*%
*%       NOTE: FOR RC/SGFB=X'0C04'-'0C10' VSE/POWER HAS RECORDED
*%             YOUR FAILING JECL STATEMENT ON THE CONSOLE WITH
*%             MSG 1Q5JI FOLLOWED BY MSG 1Q5GI !!
*%-------------------------------------------------------------------
*
* -------------------------------------------------------------------
* FOR A SUCCESSFUL IPWSEGM REQUEST, THE ATTRIBUTES OF THE CREATED
* SEGMENT ENTRY CAN BE FOUND AT '$MXSQI' IN THE MACRO PARAMETER
* AREA - WHICH IS ADDRESSED BY REG. 1 AND DESCRIBED BY $MXDS DSECT.
* THIS CODE SUGGESTS TO IDENTIFY THE SEGMENT NAME IN A MESSAGE.
* -------------------------------------------------------------------
SGM1OK1  DS    0H
         USING $MXDS,1                  MAKE PARAM. AREA ADDRESSABLE
*        MVC   MYMSGNM,$MXJNM           COPY SEGMENT NAME TO MESSAGE
```

```
*        ...   YOUR CODE TO SET UP A 'SUCCESSFUL' MESSAGE
*        ...   YOUR CODE TO WRITE MESSAGE TO THE CONSOLE
         DROP  1                         RELEASE PARM. AREA ADDRESS
* -------------------------------------------------------------------
* CONTINUE ALSO, WHEN NO SEGMENT BEEN CREATED (e.g. NOTHING SPOOLED)
* -------------------------------------------------------------------
SGM1OK2  DS    0H
* -------------------------------------------------------------------
* NOW PASS THE JOB CONTROL AND USER STATEMENTS FOR THE NEW VSE/POWER
* JOB TO BE CREATED.
* -------------------------------------------------------------------
         LA    1,CCB                     POINT TO THE CCB
         EXCP  (1)                       AND ISSUE THE SVC0
         WAIT  (1)                       AND WAIT FOR I/O TO COMPLETE
* -------------------------------------------------------------------
* AND FINALLY, PASS SECOND IPWSEGM REQUEST FOR THE PUNCH DEVICE
* TO HAVE THE NEW JOB ADDED TO THE READER QUEUE WITH UNIQUE
* EXECUTION CLASS AND JOB-NAME. AT THE SAME TIME RE-ESTABLISH
* DEFAULT PUNCH OUTPUT CHARACTERISTICS, USING NO
* EXPLICIT * $$ PUN JECL STMT. FOR THE IPWSEGM CALL
*                                                  >>> SEE NOTE 2 -->
* -------------------------------------------------------------------
*
         IPWSEGM DEVADDR=SYS008
*
         LTR   15,15                     SEGMENT CREATED SUCCESSFULLY,
*                                        - WITH RC/SGFB=X'0000'?
         BZ    EOJ                       YES, DONE, GOTO EOJ
         B     SGMERROR                  GO & IDENTIFY ANY OTHER CODES
*                                        BY MSG - SHOULD NOT HAPPEN
*                                        AFTER PROGRAM PUNCHING !!
* -------------------------------------------------------------------
* MAKE HEX VALUES OF RC/SGFB READABLE AS DECIMAL HEX REPRESENTATIONS,
* EXPECTING CODES IN REG 0. REPORT FAILURE BY MESSAGE TO CONSOLE.
* -------------------------------------------------------------------
*
SGMERROR DS    0H
         SRDL  0,8                       GET RC IN REG 0 RIGHT SIDE
         SRL   1,24                      GET SGFB IN REG 1 RIGHT SIDE
         LR    3,0                       COPY CODE TO INPUT REGISTER
         LA    4,RCD                     POINT REG. 4 TO MSG AREA
         BAL   6,HEXCONV                 GO & CONVERT 'RC'
         LR    3,1                       COPY SGFB TO INPUT REGISTER
         LA    4,SGFB                    POINT REG. 4 TO MSG AREA
         BAL   6,HEXCONV                 GO & CONVERT 'RC'
         LA    1,CCB2                    AND INFORM THE OPERATOR
         EXCP  (1)                       VIA A CONSOLE MESSAGE
         WAIT  (1)                       WAIT FOR THE I/O TO COMPLETE
         B     EOJ                       GO & TERMINATE
*
* -------------------------------------------------------------------
* HEX CONVERSION SUBROUTINE - ONE BYTE TO TWO EBCDIC BYTES
* -------------------------------------------------------------------
*INPUT: REG. 3  =  INPUT BYTE TO CONVERT
*       REG. 4  =  POINTER TO OUTPUT AREA (2 BYTES)
*       REG. 6  =  LINK REG.
*USES:  REG. 2
*
HEXTBL   DC    C'0123456789ABCDEF'       HEX-> CHAR.-HEX. REPRESENT.
         SPACE
HEXCONV  DS    0H
         SLDL  2,28                      SHIFT LEFT HALF-BYTE TO REG2
         STC   2,0(4)                    STORE IT TO OUTPUT + 0
         SRL   3,28                      SHIFT RIGHT HALF-BYTE TO R3
         STC   3,1(4)                    STORE IT TO OUTPUT + 1
         TR    0(2,4),HEXTBL             TRANSLATE OUTPUT
         BR    6                         RETURN
```

```
         SPACE
EOJ      DS    0H
         EOJ                           RETURN TO JOB CONTROL
         EJECT
* ---------------------------------------------------------------------
*    * $$ PUN CARD WITH CLASS=7, DISP=I, AND JNM=NEWJOB2
* ---------------------------------------------------------------------
*
PUNCARD  DC    CL71'* $$ PUN CLASS=7,DISP=I,JNM=NEWJOB2'
*
* ---------------------------------------------------------------------
* IPWSEGM MACRO ERROR MESSAGE TEXT
* ---------------------------------------------------------------------
MSG1     DC    C'IPWSEGM MACRO RETURN/FEEDBACK CODE IS X'
RCD      DC    CL2' '
SGFB     DC    CL2' '
         DC    C'
MSG1LN   EQU   *-MSG1                  DYNAMIC MESSAGE LENGTH
         SPACE
RCFB0004 DC    AL1($MXR00)             RC CONSTANT X'00'
         DC    AL1($MX00IG)            SGFB CONSTANT X'04'
MAX18    DC    AL1($MX00NE+X'04')      CONSTANT X'18', HIGHER THAN
*                                      EXPECTED HIGHEST X'14'
         DS    0D
* ---------------------------------------------------------------------
* CCB AND CCW FOR CONSOLE I/O
* ---------------------------------------------------------------------
CCB2     CCB   SYSLOG,CCWADDR2
CCWADDR2 CCW   09,MSG1,X'20',MSG1LN
* ---------------------------------------------------------------------
* CCB AND CCWS FOR PUNCHING JCL AND USER STATEMENTS
* ---------------------------------------------------------------------
CCB      CCB   SYS008,CCWADDR
CCWADDR  CCW   01,BUF02,X'60',X'0050'
         CCW   01,BUF03,X'60',X'0050'
         CCW   01,BUF04,X'60',X'0050'
         CCW   01,BUF05,X'60',X'0050'
         CCW   01,BUF06,X'60',X'0050'
         CCW   01,BUF07,X'60',X'0050'
         CCW   01,BUF08,X'20',X'0050'
* ---------------------------------------------------------------------
* CONSTANTS FOR JOBSTREAM BEING PUNCHED
* ---------------------------------------------------------------------
BUF02    DC    CL80'// JOB NEWJOB2'
BUF03    DC    CL80'// PAUSE'
BUF04    DC    CL80'// EXEC LIBR'
BUF05    DC    CL80'A S=IJSYSRS.SYSLIB'
BUF06    DC    CL80'LD IPW$$NU.PHASE'
BUF07    DC    CL80'/*'
BUF08    DC    CL80'/&&'
         END
/*
// EXEC LNKEDT
// ASSGN SYS008,SYSPCH
// LIBDEF PHASE,SEARCH=IJSYSRS.SYSLIB
// EXEC SEGMTEST
/&
```

Chapter 4. Dynamic Access to VSE/POWER Job Attributes

Whenever during processing of a VSE/POWER job the executing program requires information about the attributes of the active VSE/POWER job, you may call the GETFLD Assembler macro from your program.

The following example shows how to code the GETFLD macro call and how to address and find the following:

- The name of the active VSE/POWER job
- Its start time
- Its VSE/POWER job number
- The name of the user who submitted the job
- The contents of the * $$ JOB UINF='...' operand (formerly USER=)

```
         ...   ...
         SPACE 1
* ---------------------------------------------------------------------
* REQUEST ADDRESSABILITY TO THE VSE/POWER POWJOB AREA CALLING MACRO
* 'GETFLD.A' OF SUBLIB PRD1.MACLIB.
* MACRO GETFLD USES THE FOLLOWING REGISTERS:
*  - 0
*  - 15 RETURN CODE =  0, IF REQUEST OK
*                   ¬= 0, IF REQUEST FAILED
*  - 1  RETURNED POINTER TO THE POWJOB AREA
* ---------------------------------------------------------------------
         SPACE 1
         GETFLD FIELD=POWJOB           ACCESS POWJOB AREA
         SPACE 1
         USING PJBADR,1                MAKE POWJOB AREA ADDRESSABLE
         MVC   OWNPNAME,PJBPNAME       COPY VSE/POWER JOB NAME
         MVC   OWNJOBUS,PJBJOBUS       COPY UINF/USER='...' INFORMATION
         DROP  1                       RELEASE POWJOB ADDRESSABILITY
         EOJ
* ---------------------------------------------------------------------
*                  LOCAL STORAGE FIELDS
* ---------------------------------------------------------------------
OWNPNAME DS    CL8                     COPIED VSE/POWER JOBNAME
OWNJOBUS DS    CL16                    COPIED VSE/POWER USER= INFO
         SPACE 1
* ---------------------------------------------------------------------
* DESCRIBE THE JOB RELATED FIELDS OF THE POWJOB AREA BY AN OWN DSECT.
* USE THE SAME NAMES AS THE z/VSE MAPPING MACRO 'MAPPOWJB.A' OF
* SUBLIB PRD2.GEN1.
* NOTE: IF NO VSE/POWER JOB IS ACTIVE, THE POWJOB AREA CONTAINS HEX 0
* ---------------------------------------------------------------------
PJBADR   DSECT                         LAYOUT OF THE POWJOB AREA
PJBPNAME DS     CL8                    NAME OF ACTIVE VSE/POWER JOB,
*                                      - LEFTBOUND, PADDED WITH BLANKS
PJBPTIME DS     CL8                    START TIME OF ACTIVE POWER JOB,
*                                      - STORE CLOCK (STCK) VALUE
         DS     CL12                   INTERNAL JOB INFORMATION
PJBPNUM  DS     H                      NUMBER OF ACTIVE VSE/POWER JOB,
*                                      - IN BINARY FORMAT
         DS     CL6                    INTERNAL JOB INFORMATION
PJBPUSER DS     CL8                    'FROM' USER-ID OF VSE/POWER JOB,
*                                      - LEFTBOUND, PADDED WITH BLANKS,
*                                      - ALL BLANK, IF LOCALLY READ JOB
PJBJOBUS DS     CL16                   UINF/USER='..' INFO OF VSE/POWER JOB,
*                                      - LEFTBOUND, PADDED WITH BLANKS,
*                                      - ALL BLANK, IF NOT SPECIFIED
```

```
PJBPTKN  DS     F                      TKN VALUE IN BINARY FORMAT
* ------ END OF VSE/POWER JOB INFORMATION -----------------------------
         SPACE 1
         END
```

Chapter 5. Support of the IBM 4248 Printer

A program that writes to an IBM 4248 printer operating in native mode can run under control of VSE/POWER. In general, there is no need for you to change your programs. VSE/POWER handles IBM 4248-specific I/O requests as described in *VSE/POWER Administration and Operation*, SC34-2625.

As far as user-written channel programs are concerned, some of the IBM 4248-specific I/O commands cannot be processed so that they achieve the expected results. These commands are listed in Table 17.

Table 17. VSE/POWER Action for IBM 4248-Specific I/O Commands

Command		**Action by VSE/POWER During**	
Name	**Code**	**Job Execution**	**Printing Spooled Output**
Read Band ID	X'0A'	Returns the requested bytes with all bits set to zero.	Ignores the command.
Execute Order	X'33'	Spools the command, except a Purge Buffered Data order.	Ignores the command if: - Horizontal-copy printing is not set in the FCB. - The command is a Purge Buffered Data order.
Load FCB	X'63'	Spools the command, including the FCB image.	Passes the command and the image to the printer. However, VSE/POWER looses control over the printer's FCB.
Sense ID	X'E4'	Returns the requested 7-byte device ID.	Ignores the command.
Sense Intermediate Buffer	X'14'	Returns the requested bytes with all bits set to zero.	Ignores the command.
Verify Band ID	X'F3' or X'FB'	Returns the requested bytes with all bits set to zero.	Ignores the command.
Printer control commands not listed here are handled by VSE/POWER in the same way as in the past.			

Part 2. Spool-Access Support

Chapter 6. Introduction to Spool-Access Support

Spool-access support allows a program running under or outside the control of VSE/POWER to access VSE/POWER services. A program using the support can:
- Retrieve queue entries from the local VSE/POWER queues
- Submit jobs or output data for spooling to the VSE/POWER queues and retrieve VSE/POWER generated messages
- Submit control requests or pass VSE/POWER commands (such as PALTER, PDISPLAY, PHOLD, PXMIT) to control the handling of queue entries.

Normally, IBM-supplied components use this support without your noticing it.

To make the description more easily understandable and ease the entry in your own code, a sample routine has been made available (see Chapter 13, “Spool-Access Support Programming Example,” on page 271).

For this chapter, we advise that you have a copy of the *VSE/POWER Administration and Operation*, SC34-2625 publication at hand. This publication also discusses Spool Access Protection, which can limit access by user ID.

Spool-Access Support Overview

To use the available VSE/POWER-access services, your program must:
1. Set up a communication path to VSE/POWER
2. Issue one or more requests to obtain the desired spool-access service
3. Remove the existing communication path when there is no further need for access services.

You do this with the spool-access macros shown below.

XPCC requests a spool-access service by VSE/POWER. Normally, you issue several such requests in your program for a queue entry retrieval or a job or output submission; but it may also be just one request for a control-type service.

XPCCB
builds the control block (called XPCCB) needed to process an XPCC macro.

MAPXPCCB
builds a DSECT for access to an XPCCB. In this chapter, references to XPCCB-related fields or codes use the mnemonics that you find also in the generated DSECT.

PWRSPL
builds a parameter list used to pass to VSE/POWER the control information needed for the access service. VSE/POWER needs this list when your program issues the first (or only) request.

On request, the macro generates a DSECT of the SPL. In this chapter, references to SPL-related fields or codes use the mnemonics that you find also in the generated DSECT.

For a full description of these macros, see Chapter 12, “Spool-Access Support Macros,” on page 211.

Figure 2 shows how the macros XPCC, XPCCB, and PWRSPL, relate to each other; it shows how the associated control blocks and areas are used for setting up an access to VSE/POWER services.

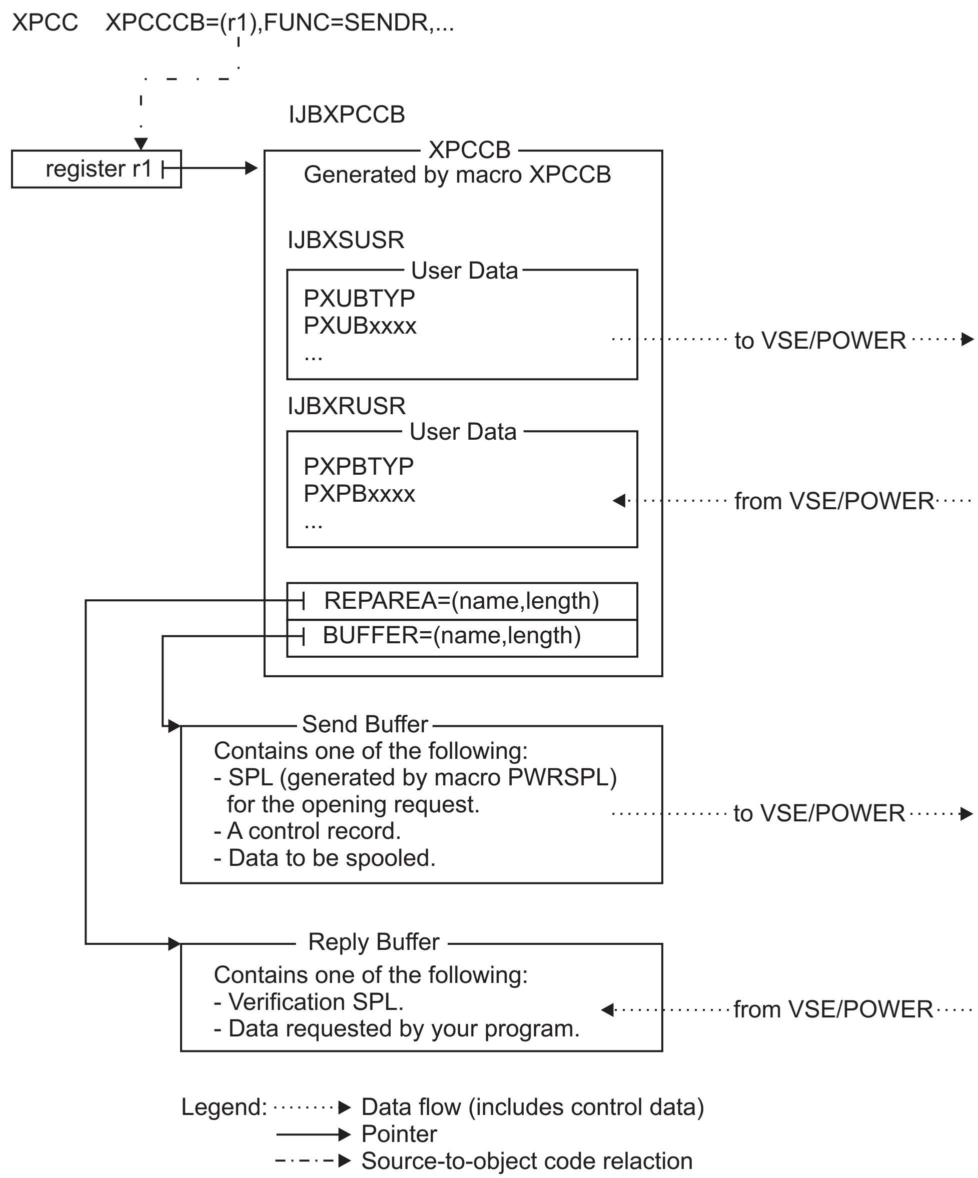


Figure 2. The Macros and Control Blocks for Spool-Access

When your program issues a service request, the system passes the associated XPCCB and send buffer to VSE/POWER. The system returns to your program's XPCCB user data passed by VSE/POWER; it puts data into your program's reply buffer as applicable.

Before your program issues a request, it must ensure that the preceding request (if any) is complete.

Separate chapters deal with setting up a communication path, issuing access-service requests, and removing an existing communication path. In studying these, you may find it helpful to have an output listing for an assembly of the following macros:

```
MAPXPCCB
PWRSPL  TYPE=MAP
```

The assembler produced DSECTs include explanatory comments.

A complete list of the VSE/POWER return and feedback codes is given in the DSECT PXPUSER, which the assembler generates for a PWRSPL TYPE=MAP macro. You find the return codes at label PXPRETCD and the feedback codes at label PXPFBKCD. For details, refer to "Spool-Access Support Parameter List (PWRSPL DSECT)" on page 231.

Setting Up a Communication Path

Sequence of Coding

To set up a communication path between your program and VSE/POWER, include in your program coding in the sequence as shown in Table 18.

For all access-service requests via an existing path, your program must use the XPCCB which you supplied for program identification and connection. The section Chapter 13, "Spool-Access Support Programming Example," on page 271 includes an identify and a connect coding sequence at labels IDENT and CONCT, respectively.

For every additional communication path established to VSE/POWER, the connection XPCCB control block must be copied from the original identification XPCCB. This is described under "Setting Up Several Communication Paths" on page 64.

Return Information

Your program should test return information as follows:

1. Register 15
2. The return code in the XPCCB field IJBXRETC

   For a complete list of possible return codes, see "XPCC" on page 212.

When the setup of the communication path is complete, your program can issue access-service requests.

Table 18. Setting Up a Communication Path Sequence

Coding in your application program	Comments
... ...	
XPCC FUNC=IDENT,... Check the return codes in register 15 and in the XPCCB (byte IJBXRETC).	Identifies your program to the system. (Required only once per program.)

Table 18. Setting Up a Communication Path Sequence (continued)

Coding in your application program	Comments
XPCC FUNC=CONNECT,... Check the return codes in register 15 and in the XPCCB (byte IJBXRETC).	The macro must refer to the same XPCCB you used for program identification. (Required for every communication path to VSE/POWER.)
WAIT IJBXCECB	Wait for the CONNECT ECB to be posted.
... ...	

Requesting VSE/POWER Access Services

You can access VSE/POWER for service requests as follows:

CTL (control) service: one or more requests to pass a command to VSE/POWER and to retrieve the message(s) produced as command responses. For details on general commands, see Chapter 7, "CTL - Passing a Command," on page 65, and for details on queue manipulation commands (selecting by the previously known queue entry number) see "Direct Queue Entry GET Access to the RDR/LST/PUN/XMT Queues" on page 79.

GET (retrieve spooled data) service: requests retrieval of a queue entry from the specified VSE/POWER

- RDR/LST/PUN queue. For details on the general variety of functions, see Chapter 8, "GET - Retrieving a Queue Entry," on page 75, and for details on specific functions (selecting by the previously known queue entry number), see "Direct Queue Entry GET Access to the RDR/LST/PUN/XMT Queues" on page 79.
- XMT queue. For details on this specific function (selecting by the previously known queue entry number), see "Direct Queue Entry GET Access to the RDR/LST/PUN/XMT Queues" on page 79.
- CRE (Create) queue. For details on this specific function (selecting by the previously known queue entry number), see "Direct GET BROWSE Access To Output Queue Entries In Creation" on page 81.

PUT (submit job or output) service: requests to include into the applicable VSE/POWER queue a job (or job stream) or output data. For more information, see Chapter 9, "PUT - Submitting a Job, a Job Stream, or Output," on page 103.

GCM (get job completion messages) service: requests to retrieve job completion, job generation, and output generation messages for jobs passed to VSE/POWER, and requests to generate and extract extended event messages which inform about VSE/POWER queue entries creation, alteration, and deletion. For more information, see Chapter 10, "GCM - Retrieving Job Event and Output Generation Messages, Inquiring eXtended Event Messages," on page 139.

Via an existing communication path, only one type of service processing can be handled at a time. You cannot, for example, open GET-service processing and issue a CTL-service request before the previously started GET processing is finished. For all requests which your program issues via the communication path, it must use the same XPCCB.

You define a request, and also control information needed by VSE/POWER, primarily in a PWRSPL-generated SPL; to some extent, you specify control information in the user area of the XPCCB or in a separate control record.

The requests for a desired service have to be coded in a certain sequence depending on the type of service. This sequence is shown in the form of a diagram followed by a discussion of the various requests.

Scope of GET/CTL Access to Queue Entries

Limitation by User ID (and Node ID)

Different rules apply to queue access depending on whether Spool Access Protection is active (refer to *VSE/POWER Administration and Operation*, SC34-2625):

If Spool Access Protection Is Not Active

If Spool Access Protection is not active, spool-access support users (as opposed to the central operator) are *only* allowed to access/manipulate job/output entries which they have created, that is, whose origin is their *node ID* and *user ID*, or to access/manipulate output entries whose destination is their *node ID* and *user ID.* Therefore, the USERID= is a mandatory parameter of the PWRSPL macro for PUT, GET, and CTL requests.

Note: When spool access users enter a PDISPLAY command via the CTL request, they can see the same job/output entries as are presented to the central operator.

The scope of GET-retrieval and CTL-manipulation access extends to:

1. locally read-in jobs, when an origin user ID has been assigned to them by the FROM= parameter of the * $$ JOB statement.
2. jobs received at their final destination via the network, that still contain a * $$ JOB statement with a FROM= parameter. Such jobs have never been processed by VSE/POWER, that is, their origin is a non-PNET system.
3. output queue entries free for general access when their destination user ID is "ANY" (GET access only). "ANY" indicates that VSE/POWER may make the output available to any user.

   Note: CTL requests to manipulate a queue entry with a destination user ID of "ANY" are permitted only by the origin user of the entry.

If Spool Access Protection Is Active

This mode of security protection can be activated when starting VSE/POWER if it was also enabled at IPL. It limits *eligible* spool entry access to *authenticated* users or programs, or to system administrators, i.e., when access is restricted to certain user IDs, these must be authenticated. Authentication requires a security logon with a password or a system component logon, such as IUI. This mode applies when using GET, CTL, or PUT OUTPUT-APPEND/RESTART, as discussed in the following sections. The same rules of access apply as when Spool Access Protection is not active, with the following differences:

- The originator's access user ID (as specified in the PWRSPL field SPLGUS) is replaced by the security logon user ID.
- Output queue entries with a destination user ID of "ANY" will be restricted to any *authenticated* user ID. If access is meant to further include non-authenticated user IDs, SECAC=NO should additionally be specified for the output entry.

PNET Considerations

A PXMIT command routed to another node via CTL will have the origin user ID replaced with the issuer's security user ID if Spool Access Protection is active, replacing the originator user ID identified in the PWRSPL field SPLGUS.

If a PXMIT command is issued by a non-authenticated user, this is indicated in the command when it is routed to the target node. PXMIT commands from systems without the Spool Access Protection feature active (e.g., downlevel systems or non-VSE systems) will be assumed to be authenticated.

Limitation by Password

The PWD= is an optional parameter of the PWRSPL macro for all spool access requests. If a queue entry is protected by an explicit password (different from internal default local value of binary zeros or default programmed access value of blank), then spool-access GET/CTL service requests must specify this password, otherwise the request is rejected. Non-matching internal default values of the password do not limit the access!

Unlimited Access

Unlimited Access for Subsystem

Only selected IBM subsystems have the capability, such as the central operator has, to gain GET/CTL access to **all** queue entries.

Unlimited Access by Installation-Specific Master Password

Each installation can define its own general password with the master password support. If this password is used in the spool-access support GET/CTL request, it provides access to all queue entries, regardless of mismatching userid and optional password.

If the master password is specified in a PUT OPEN RESTART or APPEND request, additional password checking is ignored, but the userids must match.

The master password also allows issuing commands which are for authorized users only. For information, please see the COMMAND operand of the PWRSPL macro in the topic “Format 3: Generating a DSECT” on page 219. The master password is saved in enciphered format and is, therefore, not readable in a dump.

Limitation by Maximum Number of Users

The initiation of a Spool Access task GET/PUT/CTL/GCM-OPEN request is not done by an operator, but instead by a XPCC application programs that issue XPCC CONNECT requests to SYSPWR. For each connection established, a SAS task is created in VSE/POWER. Due to logic error, a XPCC application program may loop on CONNECTing to SYSPWR without performing a DISCONNect and hence ever more SAS tasks are created until finally either partition Getvis or SETPFIX LIMIT storage is used up in the VSE/POWER partition. To aid in isolating such failures, VSE/POWER starts up with a default threshold value (MAXSAS=250) of the maximum number of concurrently active SAS tasks. When the threshold is exceeded, further XPCC CONNECT requests are terminated by XPCC DISCPRG accompanied by VSE/POWER's error return PXPRETCD/PXPFBKCD=X'10/07'. At the same time the operator is warned by message 1Q3JA. For details and for modifying the default threshold value refer to the PVARY command Format 5

'Varying the Maximum Number of SAS Tasks' in the *VSE/POWER Administration and Operation*, SC34-2625.

Ending Access to VSE/POWER Services

To end accessing VSE/POWER services via a communication path, this path is to be removed. This can be done either by a request from your program or by a request from VSE/POWER. A request from your program is indicated when there is no need for further access requests via a certain communication path and VSE/POWER has finished processing the last access request.

End of Access Requested by Your Program

Coding Sequence

Refer to Table 19, a coding-sequence diagram for the removal of a communication path from within your program. The section Chapter 13, “Spool-Access Support Programming Example,” on page 271 includes a disconnect and terminate coding sequence at labels DISCT and TERMN, respectively.

Checking the Return Information

Your program should check the return codes set by the system on completion of the XPCC macro request. This ensures an orderly termination processing. These macro-return codes are listed and briefly described under “XPCC” on page 212.

Table 19. End Access to VSE/POWER Sequence

Coding in your application program	Comments
... ...	
WAIT IJBXSECB	Wait for the SENDR ECB to be posted after your program's last access-service request.
Check the VSE/POWER return and feedback codes (in the XPCCB bytes PXPRETCD and PXPFBKCD, respectively.)	
Disconnect request XPCC FUNC=DISCONN,...	Following this request, the communication path set up in your program by XPCC FUNC=CONNECT is no longer available. To set up the path again, should this be desirable, issue an XPCC request with FUNC=CONNECT.
Check the return codes in register 15 and in the XPCCB (byte IJBXRETC).	
Terminate request XPCC FUNC=TERMIN,...	This is some sort of a log off by your program. To set up the path again, should this be desirable, start out with XPCC request specifying FUNC=IDENT.
Check the return codes in register 15 and in the XPCCB (byte IJBXRETC).	This ensures orderly discontinuation of using the spool-access support.

End of Access Requested by VSE/POWER

VSE/POWER indicates this condition by return and feedback codes in field IJBXRUSR of the XPCCB. A complete list of the VSE/POWER return and feedback codes is given in the DSECT PXPUSER, which the assembler generates for a PWRSPL TYPE=MAP macro. You find the return codes at label PXPRETCD and the feedback codes at label PXPFBKCD.

Setting Up Several Communication Paths

You may, if this is desirable, have your program set up several communication paths to VSE/POWER. To do this, proceed as follows after having identified your program (by an XPCC macro with FUNC=IDENT as described in “Setting Up a Communication Path” on page 59).

1. Copy the XPCCB used for identification

   On successful completion of the request, the z/VSE system returns an X (= cross-partition) ID in field IJBXIDK of your XPCCB. The system expects an XPCCB with this ID for a subsequent XPCC request with FUNC=CONNECT. It follows then that your program needs a copy of the XPCCB with the returned cross-partition ID for every communication path which is to be set up.
2. Issue an XPCC request with FUNC=CONNECT

   The system provides a uniquely identified communication path by inserting a P (= path) ID in field IJBXPID of the XPCCB you use.

   For any additional communication path that you want to set up, issue a new XPCC request with FUNC=CONNECT. Every one of these requests must use a new copy of the XPCCB which you used for identification.

   A connect request must be complete before you can issue the next one.

Chapter 7. CTL - Passing a Command

VSE/POWER can process only one control function per CTL-service request. Open a CTL-service request in your program if you want VSE/POWER to do one of the following:

- Pass a command or a message to another node in the network
- Alter attributes of a queue entry
- Cancel a job that is being executed
- Delete a reader or an output queue entry
- Delete FCB's or messages
- Delete any information about a checkpoint taken
- Display status information about a reader or an output queue entry or a group of entries
- Display system information
- Release a job or an output queue entry
- Request queuing of fixed format job completion messages for a released job
- Place a reader or an output queue entry into the hold state
- Load a dynamic class table
- Control printing and punching of output queue entries.

For an overview of the commands accepted by a spool-access communication path, refer to "Format 3: Generating a DSECT" on page 219.

Refer to "Scope of GET/CTL Access to Queue Entries" on page 61 for a discussion of queue entry access considerations.

Refer to Table 20 on page 66, a coding sequence diagram. It shows the kind of coding you have to supply in your program and in what sequence this coding is to be. This coding is explained in the subsequent paragraphs. Chapter 13, "Spool-Access Support Programming Example," on page 271 includes a CTL-service request at label CTLA1.

Starting the CTL Service

To start a CTL service, issue a CTL-OPEN request, which requires:

- Byte PXUBTYP of the XPCCB to be set to the value equated to PXUBTSPL. This tells VSE/POWER that the send buffer contains an SPL.
- In the send buffer, an SPL set up by a PWRSPL macro with TYPE=GEN or updated by a PWRSPL macro with TYPE=UPD so that the SPL specifies the mandatory (and optional) fields for a REQ(uest)=CTL. For details refer to "PWRSPL" on page 217 and to the list of mandatory and optional operands for the CTL service in the topic "Format 3: Generating a DSECT" on page 219.
- A reply buffer set up in your program either by specifying REPAREA=(areaname,length) or by inserting the buffer's address and length into the four-byte XPCCB fields IJBXRADR and IJBXRLNG, respectively. Any messages that VSE/POWER generates are returned to your program in this buffer (see also "Retrieving Messages" on page 67). The reply buffer must be

large enough to hold at least one message and an 8-byte prefix. For the layout of the record prefix, refer to page "Spool-Access Support Parameter List (PWRSPL DSECT)" on page 231.

Processing for a CTL service may be discontinued at any time by either a QUIT request or by a new function request. For details, see "Ending the CTL Service" on page 68.

Table 20. CTL-Service Processing Sequence

Step	Coding in your application program	Comments
	... ...	
1	Open the service XPCC FUNC=SENDR,...	Your program's send buffer must contain an SPL generated (or up-dated) for processing a CTL-OPEN request.
2	Check the return codes in register 15 and in the XPCCB (byte IJBXRETC)	
3	WAIT IJBXSECB	Wait for the SENDR ECB to be posted. This indicates that VSE/POWER has finished processing the request.
4	Check the reason code (in the XPCCB byte IJBXREAS)	This reason code is provided by the XPCC support. It must not be mixed up with any reason codes provided by VSE/POWER.
5	Check the VSE/POWER return and feedback codes (in the XPCCB bytes PXPRETCD and PXPFBKCD, respectively).	Return and feedback codes inform you about the existence of the support and how your request has been processed by VSE/POWER.
6	Check for and evaluate messages returned by VSE/POWER	If messages are to be returned, then VSE/POWER passes them to your program's reply buffer (for details, refer to "Retrieving Messages" on page 67).
7	If VSE/POWER feedback code in PXPFBKCD byte of XPCC does not indicate availability of additional messages, **go to step 9**; else proceed.	VSE/POWER indicates 'End of Data' (no more available messages) by PXP00EOD feedback code.
8	Get additional messages by XPCC FUNC=SENDR,... and **return to step 2**.	Coding for this purpose is required only if the feedback code indicates that more messages are queued. No SPL need be passed for this request; your program must set a request CTL code in the XPCCB.
9	End of Service	

Retrieving Messages

VSE/POWER queues any messages that may occur while it processes the requested CTL service. It passes these messages to your program's reply buffer.

If all of the queued messages fit into your reply buffer, VSE/POWER indicates this by a return- and feedback-code combination PXPRCOK/PXP00EOD. If the generated messages do not fit, VSE/POWER passes to your program the return- and feedback-code combination PXPRCOK/PXP00OK.

In variance to the CTLSPOOL request of the spool macro support, VSE/POWER does not return the confirmation message 1R88I, if the requested command is processed successfully. Instead, a CTL service request is terminated by the return- and feedback-code combination PXPRCOK/PXP00EOD with IJBXSLN=0 (meaning "no message queued in reply buffer").

To have VSE/POWER pass messages not yet transferred, your program must:

1. Set byte PXUBTYP of the XPCCB to zero.
2. Set byte PXUACT1 of the XPCCB to the value equated to PXUATRMR.
3. Set up a null buffer (by setting field IJBXBLN to zero).
4. Issue an XPCC FUNC=SENDR request.

The coding sequence at label DSPL2 in Chapter 13, “Spool-Access Support Programming Example,” on page 271 shows how to set up a null buffer and how to issue a RETURN-MESSAGE request.

VSE/POWER deletes messages queued but not yet transferred if your program does one of the following:

- Issues another, different service request
- Issues a QUIT request
- Ends communication via the currently used path.

In case a 'PDISPLAY queue' command has been submitted by a CTL service request, VSE/POWER accumulates the display lines in an internally built list queue entry (`$SPLnnnn`, which may be seen temporarily in a PDISPLAY of the LST queue) and passes from there messages into your reply buffer. Then VSE/POWER accepts -- during message processing -- a restart request via a restart control record. See “Requesting a Restart of the GET Spool Data” on page 93.

Although generally a CTL request does not generate a Spool Access Support operation account record, a GET operation account record is created for the implicit GET request to the $SPLnnnn entry. One can suppress this account record by generating one's own VSE/POWER phase with the ACCOUNT=(...,RXSPOOL,...) operand of the VSE/POWER macro.

Ending the CTL Service

If the processing of a CTL service is to be discontinued before it is finished, you can do either of the following:

- Issue a new request, which requires an SPL to be passed to VSE/POWER.
- Issue a QUIT request. To do this:

1. Set byte PXUBTYP of the XPCCB to zero.
2. Set byte PXUACT1 of the XPCCB to the value equated to PXUATABR.
3. Set up a null buffer (by setting field IJBXBLN to zero).
4. Issue an XPCC FUNC=SENDR request.

The coding sequence at label GQUIT in Chapter 13, “Spool-Access Support Programming Example,” on page 271 shows how to set up a null buffer and how to issue a QUIT request.

Direct Queue Entry CTL Access

Provided your program addresses only a single queue entry for manipulation by the PALTER, PDELETE, PHOLD, or PRELEASE command or for a display by the PDISPLAY command, and provided the internal VSE/POWER queue record number of the desired queue entry is known to your program in advance - then you may request direct queue entry access for the CTL-OPEN Service. When accessing directly by queue record number, VSE/POWER:

- Gains in performance, because all class chain searching is bypassed
- Returns precise and program-processible return and feedback codes in case access failed, instead of returning the operator message '1R88I NOTHING TO ALTER/HOLD...'
- Provides access to one and only one queue entry, when the traditional selection criteria are not unique
- Does not build the extra internal $SPLnnnn list queue entry that accompanies a PDISPLAY request.

How to Find the Internal Queue Record Number

If, for example, your program has created the queue entry or has identified it in a free-format or fixed-format queue display request, VSE/POWER returns the internal queue record number in the following fields:

SPLXQNUM
of a PUT-OPEN, PUT-CLOSE, or PUT-SEGMENT verification SPL

SPLXQNUM
of a GET-OPEN verification SPL

QNUM
of a free-format (FULL=YES) queue display line, as a 5-digit *decimal* number

PXFMQNUM
of a fixed-format queue display record (using PWRSPL OPT=FORMAT)

PXCRQNUM
of a checkpoint response control record

$MXQNB
of the IPWSEGM return information (refer to “Format of the Macro” on page 38)

Save this queue record number for later specification in a direct CTL-OPEN request.

Starting a Direct CTL-OPEN Request

For the PALTER, PDELETE, PHOLD, PRELEASE, or PDISPLAY command set up your SPL with:

1. either mandatory search arguments as defined for PWRSPL REQ=CTL, FUNC=ALTER/DELETE/HOLD/RELEASE, refer to “Format 3: Generating a DSECT” on page 219 or by a command in operator format using PWRSPL=CTL, FUNC=COMMAND.
2. and the direct enabling features:
   - SPLXQNUM, specifying the desired queue record number as returned by VSE/POWER
   - plus flag SPLGO2QN set up in option byte SPLGOPT2, meaning "use queue record number".

Only for the named commands, VSE/POWER respects the SPLGO2QN flag, and ignores it for other commands. When the specified mandatory (queue and jobname) and optional (jobnumber and jobsuffix) search arguments do not match the attributes of the directly retrieved queue entry, VSE/POWER replies

- for PALTER, PDELETE, PHOLD, and PRELEASE return and feedback code PXPRCOKF/PXP04NOF **plus** various settings of feedback-2 code PXPFBKC2 describing 'why not found'.
- for PDISPLAY return and feedback code PXPRCOKF/PXP04DNF **plus** various settings of feedback-2 code PXPFBKC2 describing 'why not displayed'.

For details on feedback codes returned for direct CTL requests, refer to Table 22 on page 72 and Table 23 on page 73.

When the specified search arguments match the attributes of the directly retrieved queue entry, the corresponding command handles the entry as if searched and found by a non-direct CTL-OPEN request.

Respect the following attributes of direct CTL-OPEN requests:

1. 'Generic' jobname (e.g. *ABC) requests are rejected by RC/FB code PXPRCERR/PXP08GJN
2. Processing an operator type command (PWRSPL REQ=CTL, FUNC=COMMAND), additional C-type search operands are ignored.

Enabling Job Completion Messages by the Release Command

With the CTL-OPEN request for a Release command (PWRSPL FUNC=RELEASE or FUNC=COMMAND) of a job residing in the reader queue, it is also possible to ask for queueing of fixed format job completion messages, whenever the job has been processed. For that purpose one specifies in the CTL-OPEN SPL:

- flag SPLGFB2.SPLGF2MR — release to trigger completion message
- flag SPLGFB1.SPLGF1QM — queue job completion message

The completion message is queued for a fixed format message queue identified by XPCC_applid and Spool Access userid (SPLGUS) of the Release CTL request.

These completion messages can be requested for jobs spooled to the reader queue either:

- from other input sources than Spool Access PUT-JOB, or
- from PUT-JOB without 'queue job event message' options.

The 'completion message for a release' is only issued once, this means at the processing completion time of the released job. When the released job has been submitted by PUT-JOB with 'queue job event messages' options, then an additional 'completion message for the submitter' is also queued provided the message target is different from the applid.userid of the Release CTL request. For retrieval of the queued messages refer to Chapter 10, “GCM - Retrieving Job Event and Output Generation Messages, Inquiring eXtended Event Messages,” on page 139.

Deleting Checkpoint Information

The CTL request 'delete checkpoint information' allows to delete checkpoint information (including checkpoints with extended information). To set up a "delete checkpoint information" request, issue a CTL-OPEN request using the SPL fields listed below.

Table 21. SPL Fields Applicable to 'Delete Checkpoint Information' Request

Field Name	Applicability	Purpose/Contents
SPLGSRB	M	Subrequest, must be SPLGSRDC = X'08'
SPLGJB	M	Job name [1]
SPLGJN	M	Job number
SPLGJS	O	Job suffix
SPLGPW	O	Password
SPLGUS	M	User ID
SPLGQI	M	Queue ID
SPLXQNUM	M	Queue entry number [2]
Legend: M = Mandatory; O = Optional		

Table 21. SPL Fields Applicable to 'Delete Checkpoint Information' Request (continued)

Field Name	Applicability	Purpose/Contents
[1] If the job name consists of less than 8 characters, the job name must be padded with blanks at the end. You can not specify a generic job name. [2] For this request, the queue entry number must be contained in the field SPLXQNUM of the SPL. The queue entry number is not displayed by any VSE/POWER commands. The queue entry number can be obtained in one of the following ways: • The OPEN request of a GET service returns a verification SPL. It contains the queue entry number within field SPLXQNUM. • When a checkpoint is *recorded*, a checkpoint-response control record is returned. This record contains the queue entry number within field PXCRQNUM. • When a checkpoint with extended information is *retrieved*, a checkpoint-response control record is returned. This record contains the queue entry number within field PXCRQNUM. • When status information of a queue entry is displayed in fixed format (using the CTL service with the parameter OPT=FORMAT), the queue entry number is contained within field PXFMQNUM. The queue entry number is used to identify the job of which the checkpoint information is to be deleted. If the queue entry number identifies a job of which the job name and/or job number and/or job suffix are not the same as specified in the SPL fields, the return and feedback codes PXPRCOKF/PXP04NOF (X'04'/X'01') are returned indicating 'job not found'. Then feedback-2 code PXPFBKC2 can be used to clarify 'why not found'. See Table 23 on page 73 for possible feedback-2 codes.		

The CTL request 'delete checkpoint information' does not return any message, but only a return and feedback code.

Checking the Return Information for CTL Service Requests

For the return information to be checked by your program after an XPCC request, refer to “XPCC” on page 212.

Your program should also check the return codes from VSE/POWER. Provide for this checking after your program's SENDR ECB has been posted.

Table 22 on page 72 lists the return and feedback codes that VSE/POWER may supply when it processes a CTL-service related request. The list is ordered in ascending order by code values. It relates the codes to the applicable request types and gives the names that are equated to the feedback codes. A complete list of the VSE/POWER return and feedback codes is given in the DSECT PXPUSER, which the assembler generates for a PWRSPL TYPE=MAP macro. You find the return codes at label PXPRETCD, and the feedback codes at label PXPFBKCD, and the feedback-2 codes at label PXPFBKC2.

For more information on the subject, see Chapter 14, “Return and Feedback Codes and Their Meanings,” on page 297.

Table 22. Return and Feedback Codes (PXPRETCD/PXPFBKCD) for CTL-Service Related Requests

			Request Type	
Mnemonic PXPFBKCD	**Return Code**	**Feedback Code**	**CTL-Open**	**Get Messages**
PXP00OK	00	00	X	X
PXP00EOD		01	X	X
PXP04NOF [1]	04	01	X	
PXP04BSY [1]		03	X	
PXP04SOA		09	X	
PXP04DNF [2]		0B	X	
PXP04TQN		0C	X	
PXP04NCK		15	X	
PXP08SPL	08	01	X	
PXP08REQ		02	X	
PXP08SRQ		03	X	
PXP08FB2		04	X	
PXP08JNM		05	X	
PXP08QID		06	X	
PXP08CLS		07	X	
PXP08PWD		08	X	
PXP08UID		09	X	
PXP08BTS		1A	X	X
PXP08IAB		1C		X
PXP08CON		22	X	X
PXP08IBT		24		
PXP08BOS		27	X	
PXP08JNO		31	X	
PXP08JSF		32	X	
PXP08IQN [3]		44	X	
PXP08GJN [3]		45	X	
PXP0CINS	0C	01	X	X
PXP0CIXF		02	X	X
PXP0CIOE		07	X	X
PXP0CSNF[2]		08	X	
PXP0CCOR[2]		09	X	
PXP10PSP	10	05	X	X
PXP10SIE		06	X	X
PXP10MST		07	X	

[1] This feedback code appears only for direct CTL-Service (PALTER, PDELETE, PHOLD, PRELEASE) requests or for Delete Checkpoint requests. If PXP04NOF, check PXPFBKC2 of Table 23 on page 73 for detailed reason.

[2] This feedback code appears for the PDISPLAY command only. When passed as direct CTL-Service request, check PXPFBKC2 of Table 23 on page 73 for detailed information.

[3] This feedback code appears only for direct CTL-Service requests or for Delete Checkpoint requests.

Table 23. Feedback-2 Codes (PXPFBKC2) for Direct CTL-Service Requests

			at CTL-OPEN Request for					
Mnemonic PXPFBKCD/PXPFBKC2	**Return/Fdbk Code**	**Fdbk-2 Code**	**PALTER cmd**	**PDELETE cmd**	**PHOLD cmd**	**PRELEASE cmd**	**PDISPLAY cmd**	**Delete Checkpoint**
PXP04NOF/PXPC2TEM	04/01	01			X	X		
PXPC2NOH		02			X			
PXPC2NOR		03				X		
PXPC2NTA		04	X					
PXPC2CPO		05	X					
PXPC2CDI		06	X					
PXPC2CNT		07	X					
PXPC2BAD		08	X	X	X	X		X
PXPC2FRE		09	X	X	X	X		X
PXPC2MQU		0A	X	X	X	X		X
PXPC2MJM		0B	X	X	X	X		X
PXPC2MJB		0C	X	X	X	X		X
PXPC2IPW		0D	X	X	X	X		
PXPC2BPW		0E	X	X	X	X		
PXPC2JFR		0F	X	X	X	X		
PXPC2OT1		10	X	X	X	X		
PXPC2OT2		11	X	X	X	X		
PXPC2OT3		12	X	X	X	X		
PXPC2OTN		13	X	X	X	X		
PXPC2MJS		14	X	X	X	X		
PXPC2SAC		19	X	X	X	X		
PXPC2INC		1A	X	X	X	X		
PXPC2DEL		1B	X	X	X	X		X
PXP04DNF/PXPC2BAD	04/0B	08					X	
PXPC2FRE		09					X	
PXPC2MQU		0A					X	
PXPC2MJM		0B					X	
PXPC2MJB		0C					X	
PXPC2INC		1A					X	
PXPC2DEL		1B					X	

Note: For a detailed explanation of the PXPFBKC2 mnemonics, see “Spool-Access Support Parameter List (PWRSPL DSECT)” on page 231.

Chapter 8. GET - Retrieving a Queue Entry

You request a GET service if you want VSE/POWER to retrieve a certain queue entry of a local (RDR, LST or PUN) queue and make this entry available to your program. In your program, you issue GET-service requests as follows:

1. An *Open* request to start the desired retrieval of spool data – For details, see "Starting the GET Service" on page 84.
2. One or more *GET spool data* requests to have VSE/POWER make the desired spool data available to your program – For details, see "Retrieving Spool Data" on page 86.
3. An end-service request, which may be one of the following:
   - A *Close* request to indicate that the retrieval of a specific queue entry is finished – For details, see "Issuing a CLOSE Request" on page 86.
   - A *QUIT* request to end any further retrieval of spool data – For details, see "Issuing a GET-QUIT Request" on page 87.
   - A *QUIT-and-LOCK* request to indicate, for example, that the processing of an output queue entry failed - For details, see "Issuing a QUIT-and-LOCK Request" on page 87.
   - A *PURGE* request to end any further retrieval of spool data and to purge the accessed queue entry from its queue – For details, see "Issuing a PURGE Queue Entry Request" on page 87.

You may, in addition, issue:

- A *Checkpoint* request to record a suitable restart point should a restart be desirable or become necessary – For details, see "Requesting a Checkpoint" on page 88.
- A *Restart* request to set up the retrieval of a queue entry's spool data at a point other than the beginning – For details, see "Requesting a Restart of the GET Spool Data" on page 93.
- A *Get OPTB* request to obtain one or more available output parameter text blocks (OPTBs) - For details, see "Issuing a Get-OPTB Request" on page 96.
- A *Modify OPTB* request to change an OPTB - For details, see "Issuing a Modify-OPTB Request" on page 97.

For a discussion of queue access considerations, see "Scope of GET/CTL Access to Queue Entries" on page 61.

Introduction to the GET Service

Starting the GET Service

Getting a RDR/LST/PUN Queue Entry for Update

Queue entries must have a dispatchable disposition of D (delete after processing) or K (keep after processing) in order to be selectable for retrieval by the GET service for viewing and optional update. Non-dispatchable entries (DISP=H|L), or dispatchable time event scheduling jobs, or even active queue entries (DISP=*) are **not** accessible by this GET service. Further general access limitations of the GET service are discussed in "Scope of GET/CTL Access to Queue Entries" on page 61.

For extended retrieval of generally dispatchable (DISP=D|K) and non-dispatchable (DISP=H|L|X|Y|A) and possibly even active (DISP=*) queue entries, refer to "Browsing a Queue Entry for Viewing Only" on page 78.

Service End Disposition of a Retrieved Queue Entry

If you end the retrieval of a queue entry by a CLOSE request, then this retrieval is for VSE/POWER the same as processing this entry. Therefore, if the entry's disposition was

D VSE/POWER deletes the entry.

K VSE/POWER retains the entry with the entry's disposition changed to L.

For further information on disposition, please refer to the *VSE/POWER Administration and Operation*, SC34-2625.

For a summary of allowed requests for this GET service, refer to the 'GET-SPOOL' block in Appendix C, "Spool-Access Support Graphical Description," on page 379.

Getting RDR/LST/PUN Entries for Update in Generic Mode

This mode of the standard GET service (see "Getting a RDR/LST/PUN Queue Entry for Update" on page 75) does not search for a specific jobname but rather selects the next suitable entry in the specified queue and class(es). For details and additional options, see the MODE=GENERIC operand of the PWRSPL macro.

Data Passed by VSE/POWER

If your program requests a RDR queue entry, VSE/POWER does not return the * $$ JOB, * $$ CTL, and * $$ EOJ statements.

Every record made available by VSE/POWER is preceded by an eight-byte prefix as shown in Table 24 on page 77. A DSECT of this prefix, labeled RECPRFIX, is available to you if you issue a PWRSPL macro with TYPE=MAP. For the layout of the record prefix, refer to Table 24 on page 77 and "Spool-Access Support Parameter List (PWRSPL DSECT)" on page 231.

Table 24. Record Prefix Layout

Bytes	Meaning
0	Carriage control character or X'00'.
1	Record type: X'00' = Normal data record X'01' = Spool parameter list (SPL) X'02' = Fixed format message X'03' = Separator-page (separator-card) start record X'04' = 3540 data record (applies only to a RDR queue entry) X'05' = Control-command record (such as skip to channel 1 (X'8B) or block data check (X'73') X'06' = CPDS (composed page data stream) record, always indicated when the carriage control character is X'5A' X'07' = Separator-page (separator-card) end record X'08' = End-of-copy record X'09' = Fixed format job completion message (applicable only for GCM requests). X'0A' = Fixed format job generation message (applicable only for GCM requests). X'0B' = Fixed format output generation message (applicable only for GCM requests). X'0C' = Fixed format extended event message (applicable only for GCM requests).
2-3	Length of the subsequent logical record (binary)
4-7	VSE/POWER assigned record number (binary); you can use this number to specify a restart point should a restart become necessary.

If your program requests an output queue entry with multiple data set header records (DSHR), VSE/POWER builds for each DSHR an SPL and passes this SPL (with the prefix) between data records back in the reply buffer. The RECLOGNO will not be incremented by this received SPL.

The Verification SPL

In response to your first (opening) request, VSE/POWER returns to your program a verification SPL. Consider analyzing this SPL in your program and coding programmed actions that may be necessary.

The verification SPL contains the same information as the SPL passed by your program. Some of the verification SPL's fields contain data about the currently accessed queue entry and not supplied by your program. Examples are: record format and length, number of print lines or pages. Your program may need this information for setting up output processing.

Required Buffers

Your program must provide buffers as follows:

- A send buffer for the opening request, large enough to hold the required SPL. You can define the buffer by way of the BUFFER operand of the XPCC or XPCCB macro.
- A reply buffer large enough to hold either of the following whichever is larger:
  - The verification SPL passed by VSE/POWER in response to your program's opening request
  - The largest data record of the requested queue entry
  - The largest OPTB.

  You define the buffer by way of the REPAREA operand of the XPCCB macro.

Overview of the Checkpoint and Restart Facility

Checkpoint and Restart

Checkpoint and Restart is a method of recording information about a queue entry at programmer-designated checkpoints.

If necessary, a program can request VSE/POWER to restart the retrieval of spooled records. The queue entry can be restarted at any of the checkpoints or at the beginning of a queue entry.

Usually, the record and the copy number of a queue entry can be tagged with a checkpoint.

For detailed information on checkpoint/restart, please see “Requesting a Checkpoint” on page 88 and “Requesting a Restart of the GET Spool Data” on page 93.

Checkpoint with Extended User Data Information

The 'checkpoint with extended information' is specified by the user's application program and is useful in cases where the normal checkpoint information is not sufficient. It can contain any information which is **not** checked by VSE/POWER. For example, a checkpoint with extended information can be used after a print failure when it is necessary to associate the entire printer setup with a checkpoint and be passed to the requestor in order to restart the queue entry. For detailed information, see “Requesting a Checkpoint with Extended Information” on page 89.

Ending the GET Service

Your program can end a GET service at any time after completion of a relevant XPCC SENDR request. For more information, see “Ending the GET Service” on page 86.

Browsing a Queue Entry for Viewing Only

This mode of the GET service has no access restriction imposed by the disposition of the queue entry or by the target SYSID if running shared. However, the only accepted service-end request is QUIT. For details on how to specify BROWSE mode, refer to the MODE=BROWSE operand of the PWRSPL macro.

For a summary of allowed requests for this GET service, refer to the 'BROWSE-SPOOL' block in Appendix C, “Spool-Access Support Graphical Description,” on page 379.

Parallel Browsing of Queue Entries

As described under “Scope of GET/CTL Access to Queue Entries” on page 61, concurrent GET requests to the **same** queue entry may be issued by

- System administrators via the Master Password to all entries
- General users if the output entry has the to-user destination of ANY
- General users if the entry has a different from- or to-user destination.

GET-OPEN for parallel browsing are accepted by VSE/POWER, up to a maximum number of

- 255 parallel browse requests on a non-shared system
- 15 parallel browse requests per sharing system.

If, however, this maximum number of browse requests has been reached, VSE/POWER rejects the browse request with RC/FBKD=PXPRCOKF/PXP04BSY.

You may track the number of concurrent browse requests per queue entry

1. Externally, by an operator queue display command with the option FULL=YES, which presents the MACC= (Multiple Access Count) value. For details, refer to *VSE/POWER Administration and Operation*, SC34-2625.
2. Internally, by a 'fixed format' queue display request (see the FORMAT operand of the PWRSPL macro), which presents the multiple access count(s) in the PXFMMACC area (refer to “Spool-Access Support Parameter List (PWRSPL DSECT)” on page 231).

Note:

1. VSE/POWER 6.4 and previous releases did not allow concurrent GET-OPEN for update and GET-OPEN for browse. This restriction has been removed effective with 6.5.
2. Unlike GET for update, a 'browsed' queue entry being accessed by *one or more* tasks per (shared/nonshared) CPU does not show pages/cards/lines and copies 'left to be processed' in a normal or fixed-format queue display; instead, it shows the 'total' values.

Direct Queue Entry GET Access to the RDR/LST/PUN/XMT Queues

Provided the internal VSE/POWER queue record number of the desired queue entry is known to your program in advance, then you may request direct queue entry access for the GET-OPEN Service. When accessing directly by queue record number, VSE/POWER:

- Gains in performance, because all class chain searching is bypassed
- Returns precise return and feedback codes in case access failed
- Provides access to the exact queue entry, when the standard selection criteria are not unique.

How to Find the Internal Queue Record Number

If, for example, your program has created the queue entry or has identified it in a free-format or fixed-format queue display request, VSE/POWER returns the internal queue record number in the following fields:

SPLXQNUM
of a PUT-OPEN, PUT-CLOSE, or PUT-SEGMENT verification SPL

SPLXQNUM
of a GET-OPEN verification SPL

QNUM
of a free-format (FULL=YES) queue display line, as a 5-digit decimal number

PXFMQNUM
of a fixed-format queue display record (using PWRSPL OPT=FORMAT)

PXCRQNUM
of a checkpoint response control record

$MXQNB
of the IPWSEGM return information (refer to "Format of the Macro" on page 38)

Save this queue record number for later specification in a direct GET-OPEN request.

Starting a Direct GET-OPEN Request

For a GET-OPEN for update or browse, set up your SPL with:

1. mandatory (and optional) search arguments as defined for PWRSPL REQ=GET (refer to "Format 3: Generating a DSECT" on page 219).
2. and the direct enabling features:
   - SPLXQNUM, specifying the desired queue record number as returned by VSE/POWER
   - plus flag SPLGO2QN set up in option byte SPLGOPT2, meaning 'use queue record number'.

See also Table 25 on page 83.

When the specified search arguments (as queue, jobname) do not match the attributes of the directly retrieved queue entry, VSE/POWER replies:

- the standard return and feedback code PXPRCOKF/PXP04NOF **plus** various settings of feedback-2 code PXPFBKC2, describing 'why not found'.

For details on feedback codes returned for direct GET requests, refer to Table 28 on page 98 and Table 30 on page 100.

When the specified search arguments match the attributes of the directly retrieved queue entry, the GET Services continues to respond with the verification SPL as done for a non-direct GET-OPEN request.

Observe the following restriction for direct GET-OPEN:

- Direct specifications are not respected for a generic (PWRSPL MODE=GENERIC) GET-OPEN request.

Special Considerations for Access to the XMT Queue

- For such access, specify PWRSPL QUEUE=XMT or set field SPLGQI explicitly to SPLGQIX (C'X'). This queue type, together with the R/L/P type indicated by SPLGFLG, is also returned in your Verification SPL in response to a successful GET-OPEN request.
- When you request a fixed-format display of the XMT queue, field PXFMQUID will not show C'X' for XMT queue entries but rather present the current 'R/L/P' type as for local queues. The attribute 'entry resides in XMT queue' is in fact presented by the PXFMF1XQ indication of control flag PXFMFLG1.

Direct GET BROWSE Access To Output Queue Entries In Creation

VSE/POWER offers access to queue entries in creation by *execution writer tasks*, which is the basic way of creating output. For example, the output of the CICS/ICCF job can now be browsed and analyzed while it is still in creation. Direct GET-open must be used for access, and only browsing is allowed for an entry in creation. For a shared spooling complex, the creating task and all tasks browsing the queue entry in creation must reside on the same system. These restrictions are needed:

- for access to the queue entry in creation, because only direct GET-OPEN is able to find it and select it for further processing. Such an entry is not yet chained and is tied only to the creating task.
- for data integrity, because the queue entry in creation must not be changed or deleted by any other task except the creating task. Therefore, only browsers, which never change the queue entry, are allowed to access the entry.
- for performance reasons, because the browser must read the virtual storage of the creating task (on the same shared system) to find the last spooled records.

Whenever a browser requests access to a queue entry in creation, VSE/POWER ensures that the mentioned criteria are fulfilled:

- direct GET request for BROWSE issued for queue entry in creation
- queue entry created by execution writer task
- queue entry in creation on local system

VSE/POWER furthermore ensures that all spooled records are written to disk and a temporary end of data is set. Therefore, during selection of an entry in creation a snapshot is created for the requesting browser:

- the queue record copy used by the browser is updated to reflect the current record and page counts of the entry in creation.
- the last spooled data records collected in storage are written to disk.
- a temporary end of the data is maintained for the browser.

Although the creating task may spool more records, this snapshot is never modified for the associated browser for as long as it processes the queue entry. This prevents mismatches between the record counts passed via the Verification SPL at GET-OPEN time and the actual number of records passed to the browser. To refresh the snapshot, the browser must end its processing with a QUIT request and re-open the in-creation queue entry again.

Searching for Queue Entries in Creation

The PDISPLAY command can be used with the operand **CRE** and its sub-operands to show the logical Create queue, which is the set of all queue entries being created.

- For output entries (LST/PUN) in creation, all information needed for the display has already been inherited from the creating job or defined by a *** $$ LST/PUN** statement or by a SAS PWRSPL and can therefore be shown.
- For RDR queue entries in creation, *jobname*, *class*, *disposition* and other fields may still show defaults, if for example a *** $$ JOB** statement has not yet been read.

Starting a Direct GET-OPEN for BROWSE of Queue Entries in Creation

See Table 25 on page 83. To set up the required direct GET request for BROWSE, the program must

1. set byte SPLGFB1=SPLGF1BR, meaning GET request for BROWSE
2. set up the SPL with mandatory and optional search arguments as defined for PWRSPL REQ=GET (refer to "Format 3: Generating a DSECT" on page 219). The mandatory QUEUE= specification must specify the LST or PUN queue corresponding to the value of the "I" column in message 1R4BI (free-format display line of the Create queue) or to the PXFMQUID indication of a fixed-format display record for the Create queue.
3. set the direct enabling features
   - SPLXQNUM, specifying the desired queue record number in hexadecimal format. VSE/POWER returns this number in decimal format in
     - message 1R4BI for PDISPLAY CRE
     - message 1R48I for PDISPLAY A

     The queue record number in hexadecimal format is returned in field PXFMQNUM in the fixed-format queue display.
   - flag SPLGO2QN in option byte SPLGOPT2, meaning 'use queue record number'
4. set byte SPLGOPT=SPLGOGIC, meaning GET request for queue entry in creation.
5. supply a reply buffer to which VSE/POWER passes the verification SPL.
6. issue a SENDR request.

In response to such a request, VSE/POWER will

- either reject the request with RC/FB PXP04NOF (queue entry not found) and feedback2 code:
  - PXPC2NVT - if the queue entry is either in creation on another system of a shared spooling complex, or it is in creation but the creating task does not support GET BROWSE for a queue entry in creation.
  - PXPC2EMP - if the queue entry in creation is still empty
  - PXPC2QCL/-P/-R/-X - queue entry is no longer in creation but can be found in the LST/PUN/RDR/XMT queue.
  - or another applicable feedback2 code (see Table 30 on page 100).
- or it will return the verification SPL of the queue entry in creation.

When the verification SPL has been returned, normal GET BROWSE processing takes place. There is no difference between browsing a queue entry in creation and a normal queue entry.

Only data and control records in the range defined by SPLDRCT and SPLDLCT are passed to the requesting SAS program for a Spool Data Request. When the temporary end (defined by SPLDRCT) is reached, VSE/POWER will inform the program by the return/feedback code PXP00EOF.

To read data spooled after opening the queue entry, the program must issue a QUIT request, followed by a new direct GET-OPEN for BROWSE of a queue entry in creation. If the queue entry is still in creation, a verification SPL is passed to the program, which contains the updated spooling state. If the queue entry has been completed in the interim, RC/FB/FB2=04/01/PXPC2QCL|-P|-R|-X will be returned to let the program decide how to continue.

Mandatory and Optional Operands for GET-OPEN

The following table summarizes the different settings in PWRSPL for the various types of Spool-Access Support GET-OPEN, which are described in detail in this chapter.

Table 25. Mandatory and Optional Operands for GET-OPEN

Operand	Description	OPEN for UPDATE		OPEN for BROWSE		OPEN for BROWSE in creation
		Normal	Direct	Normal	Direct	Direct
SPLGRQB = SPLGRGET (X'02')	Request Byte identifies GET request	M	M	M	M	M
SPLGFB1 = SPLGF1BR (X'03')	Function Byte 1 identifies BROWSE request			M	M	M
SPLGOPT. SPLGOGIC (X'02')	Option Byte 1, flag identifies In CREATION					M
SPLGOPT2. SPLGO2QN (X'10')	Option Byte 2, flag identifies Direct Access		M		M	M
SPLXQNUM	Internal Queue Record Number		M		M	M
SPLGQI	Queue ID					
= R/L/P		M		M		
= R/L/P/X			M		M	
= L/P						M
SPLGJB	Job Name	M	M	M	M	M
SPLGCL	Class	M	M	M	M	M
SPLGUS	User ID	M	M	M	M	M
SPLGJN	Job Number	O	O	O	O	O

Table 25. Mandatory and Optional Operands for GET-OPEN (continued)

Operand	Description	OPEN for UPDATE		OPEN for BROWSE		OPEN for BROWSE in creation
		Normal	Direct	Normal	Direct	Direct
SPLGJS	Job Suffix	O	O	O	O	O
SPLGPW	Password	O	O	O	O	O
Note: M = mandatory; O = optional						

Coding Sequence for the GET Service

Table 26 shows the kind and sequence of the coding needed in your program for the retrieval of a complete queue entry. This coding is explained in the subsequent paragraphs. Chapter 13, "Spool-Access Support Programming Example," on page 271 includes a GET-service request at label GETB1.

Starting the GET Service

To open GET-service processing, the application program uses the GET-OPEN service, which requires:

- Byte PXUBTYP of the XPCCB to be set to the value equated to PXUBTSPL. This indicates to VSE/POWER that the send buffer contains an SPL.
- An SPL as set up by a PWRSPL macro with TYPE=GEN or updated by a PWRSPL macro with TYPE=UPD so that the SPL specifies the mandatory (and optional) fields for a REQ(uest)=GET. For details refer to the PWRSPL macro "PWRSPL" on page 217.
- A reply buffer to which VSE/POWER passes the verification SPL.

Table 26. GET Service for a Complete Queue Entry Sequence

Step	Coding in your application program	Comments
1	Open request XPCC FUNC=SENDR,...	Your program's send buffer must contain an SPL generated (or updated) for processing a GET service.
2	Check the return codes in register 15 and in the XPCCB (byte IJBXRETC).	
3	WAIT IJBXSECB	Wait for the SENDR ECB to be posted.
4	Check the reason code (in the XPCCB byte IJBXREAS).	
5	Check the VSE/POWER return and feedback codes (in the XPCCB bytes PXPRETCD and PXPFBKCD, respectively).	

Table 26. GET Service for a Complete Queue Entry Sequence (continued)

Step	Coding in your application program	Comments
6	Check for and evaluate the SPL from VSE/POWER, if necessary.	VSE/POWER returns a verification SPL to your program's reply buffer if the request has been accepted and can be processed by VSE/POWER.
7	GET spool data request XPCC FUNC=SENDR,...	Your program's XPCCB must refer to a zero-length send buffer.
8	Check the return codes in register 15 and in the XPCCB (byte IJBXRETC).	
9	WAIT IJBXSECB	Wait for the SENDR ECB to be posted. VSE/POWER places the retrieved data record(s) into your program's reply buffer.
10	Check the reason code and VSE/POWER return and feedback codes (this is the same as above in steps 4 and 5).	
11	Deblock the data in the reply buffer, if necessary. If more records are to be transferred, **return to Step 7**. Else proceed.	Loop until VSE/POWER returns the feedback code PXP00EOD.
12	End-retrieval request XPCC FUNC=SENDR,...	Your program's XPCCB must refer to a zero-length send buffer.
13	Check the return codes in register 15 and in the XPCCB (byte IJBXRETC).	
14	WAIT IJBXSECB	Wait for the SENDR ECB to be posted. This ensures that the communication is free for another service request.
15	Check the reason code and VSE/POWER return and feedback codes (this is the same as above in steps 4 and 5).	
16	End of Service	

Retrieving Spool Data

After VSE/POWER has passed the verification SPL, your program eventually issues one or more GET spool data requests, each one after the preceding one has been completed. The code in your program must do the following:

1. Set byte PXUBTYP of the XPCCB to zero.
2. Set byte PXUACT1 of the XPCCB to the value equated to PXUATSDR.
3. Set up a null buffer (by setting field IJBXBLN to zero).
4. Issue an XPCC FUNC=SENDR request.

In response to a GET spool data request, VSE/POWER fills your program's reply buffer with records of the queue entry, one record behind the other. Each record contains an 8-byte prefix. You define this buffer by the REPAREA operand of your XPCCB macro; you may want to alter this definition by changing the buffer's address (in field IJBXRADR) and its length (in field IJBXRLNG).

Ending the GET Service

When your program has finished processing the data of a queue entry, it should issue one of the following requests after VSE/POWER has completed a relevant XPCC SENDR request:

CLOSE
: To have VSE/POWER dispose of the queue entry in accordance with VSE/POWER's disposition rules.

QUIT
: To return the queue entry with its original disposition.

QUIT-and-LOCK
: To indicate that the processing of an output queue entry failed.

PURGE
: To purge the queue entry from the queue.

Issuing a CLOSE Request

In your program, you normally issue a CLOSE request when VSE/POWER has completed the retrieval of the desired queue entry. However, you can issue a CLOSE request any time during the retrieval of a queue entry.

When it receives a CLOSE request, VSE/POWER handles the queue entry in accordance with its disposition rules. If the disposition is:

D
: VSE/POWER deletes the queue entry.

K
: VSE/POWER retains the queue entry with a disposition of L.

To issue a CLOSE request in your program:

1. Set byte PXUBTYP of the XPCCB to zero.
2. Set byte PXUACT1 of your XPCCB to the value equated to PXUATRQS.
3. Set up a null buffer (by setting field IJBXBLN to zero).
4. Issue an XPCC FUNC=SENDR request.

The coding in your program is similar to a QUIT request as shown at label GQUIT Chapter 13, “Spool-Access Support Programming Example,” on page 271. However, the MVI instruction that sets byte PXUACT1 of the XPCCB is to be replaced by the sample instruction shown as comment with the label *GCLOSE.

Issuing a GET-QUIT Request

You can do this at any point during the retrieval of a queue entry. The request causes VSE/POWER to retain the queue entry with its originally assigned priority and disposition.

To issue a QUIT request in your program:

1. Set byte PXUBTYP of the XPCCB to zero.
2. Set byte PXUACT1 of your XPCCB to the value equated to PXUATABR.
3. Set up a null buffer (by setting field IJBXBLN to zero.
4. Issue an XPCC FUNC=SENDR request.

The coding sequence at label GQUIT in Chapter 13, “Spool-Access Support Programming Example,” on page 271 shows how to set up a null buffer and how to issue a QUIT request.

Issuing a QUIT-and-LOCK Request

You can do this at any point during the retrieval of a queue entry. The request causes VSE/POWER to re-queue the currently processed job in the appropriate non-dispatchable class chain with a temporary disposition of Y for the purpose of:

- Indicating that a problem has occurred during output processing, and
- Preventing that the output queue entry is handled again until the subsystem has taken some special action (for example, issued the PALTER command to alter the temporary disposition to a dispatchable one). For details on disposition Y handling, see “Handling an Abnormal-End Condition During GET” on page 100.

To issue a QUIT-and-LOCK request in your program:

1. Set byte PXUBTYP of the XPCCB to zero.
2. Set byte PXUACT1 of your XPCCB to the value equated to PXUAT1PF.
3. Issue an XPCC FUNC=SENDR request passing a null buffer, that is, a buffer with a length of zero (IJBXBLN set to zero).

Issuing a PURGE Queue Entry Request

You can do this at any point during the retrieval of a queue entry. The request causes VSE/POWER to delete the currently processed queue entry from its queue.

To issue a PURGE request in your program:

1. Set byte PXUBTYP of the XPCCB to zero.
2. Set byte PXUACT1 of your XPCCB to the value equated to PXUATPRG.
3. Issue an XPCC FUNC=SENDR request passing a null buffer, that is, a buffer with a length of zero (IJBXBLN set to zero).

The coding in your program is similar to a quit request as shown at label GQUIT in Chapter 13, “Spool-Access Support Programming Example,” on page 271. However, the MVI instruction that sets byte PXUACT1 of the XPCCB is to be replaced by the sample instruction shown as comment with the label *GPURGE.

Converting ASA Characters to Machine Control Characters

You may have found out by a PDISPLAY CTL-service request that a certain output entry contains ASA control characters. This is indicated either by the record format (RF) field of a FULL=YES display request, or by the record format field of a fixed format queue display request. A GET-service request offers the ASA controlled data records unchanged to your program.

However, you can ask VSE/POWER to do ASA to machine control conversion by setting option byte SPLGOPT2 of the GET-service-open SPL to SPLGO2AC. Then VSE/POWER passes - for every list type ASA data record - two machine control records to your program:

- a first one doing the forms control operation
- a second one writing the actual data immediately.

The VSE/POWER assigned record number contained in the record prefix is the same for both generated machine control records, since they stem from one ASA record. Punch type ASA records are not split into two during conversion. Instead, their ASA operation code is changed to X'00', leaving it up to your program to select an operation code that is punch device specific.

Requesting a Checkpoint

Checkpointing is meaningful if your program requests a large amount of spooled data to be retrieved. It is meaningful, for example, if your program is to process retrieved spool data in sections. It can save processing time should a program or system failure occur.

Your program can request VSE/POWER to record a checkpoint at any time between two GET spool data requests. VSE/POWER records checkpoint information as follows:

- Logical record number as specified in the checkpoint-control record.
- The output-copy number (if applicable).

To have VSE/POWER record a checkpoint, your program must:

1. Set up a checkpoint-control record in your program's send buffer.

   By issuing a PWRSPL macro with TYPE=MAP, the assembler generates a DSECT of this record at label PXCPDSCT.

   In the checkpoint-control record, you can specify a copy number (field PXCPRCPY) if the control record applies to an output queue entry. The number tells VSE/POWER, that checkpoint information is to be recorded for the specified record in the specified output copy. If you set the field to zero, VSE/POWER uses its current number-of-copies count.
2. Set byte PXUACT1 of the XPCCB to zero.
3. Set byte PXUBTYP of the XPCCB to the value equated to PXUBTCTL.

   This tells VSE/POWER your program's send buffer contains a control record.
4. Issue an XPCC FUNC=SENDR request.

   The request passes to VSE/POWER the checkpoint-control record which your program has set up in its send buffer.

After having recorded the requested checkpoint, VSE/POWER returns a checkpoint-response record (in your program's reply buffer). The assembler generates a DSECT of this record at label PXCRDSCT if you issue a PWRSPL macro with TYPE=MAP.

As described in "Requesting a Restart of the GET Spool Data" on page 93, VSE/POWER returns the last recorded checkpoint of a queue entry when a retrieval of this queue entry is started again. In your program, you can then decide whether VSE/POWER is to continue retrieval at that checkpoint (by issuing a restart request) or from the beginning (by issuing a GET spool data request).

Note: Checkpoint information can *not* be requested if the GET Service is used with the BROWSE option.

Requesting a Checkpoint with Extended Information

VSE/POWER will record the following information when a checkpoint control record with extended checkpoint information is passed:

- Logical record as specified by the user program
- Copy number associated with the logical record number
- Extended information as passed by the user program.

Processing of a Checkpoint with Extended Information

The spool-access support user passes the checkpoint with extended information by the checkpoint control record along with the record and copy number. The checkpoint control record must be flagged to indicate that the record contains a checkpoint with extended information. A checkpoint-response control record is returned to the spool-access support user which indicates that a checkpoint with extended information was taken. If recording of the checkpoint was unsuccessful, a return and feedback code is returned (see "Checking the Return Information" on page 92).

The extended information of a checkpoint is written to the VSE/POWER data file. If no spool space is available to write the extended checkpoint information onto the data file, message 1Q38I will be sent to the operator console. The spool-access support user recording the extended checkpoint information is put in the wait state until the necessary spool space becomes available.

Queue Control Area (QCA)

The extended information of a checkpoint is written into a separate area of the data file, the so-called Queue Control Area (QCA). The queue control area contains:

- Control information which is sent from one system to another system in a shared environment
- Extended information for a checkpoint.

No information is required concerning the allocation of the queue control area. The QCA is dynamically built; it uses as many DBLKs of the data file as are needed at the time. Whenever an I/O error occurs during accessing the queue control area, or reading the master record during a warm start all data within the queue control area is lost, which means the extended information of checkpoints of all queue entries is lost. Such a loss is indicated to the application program within the verification SPL and within the checkpoint control record.

Output Exit Routines

If output exit routines are active, these routines usually get control for every record passed to a printer or to programs controlling a printer. Such an output exit routine will not receive any information on a checkpoint with extended information.

Omission of Processing the Extended Checkpoint Information

When a queue entry is processed by any other task than a spool-access support or a device service task, the extended information of a checkpoint is **not** processed. For example:

1. If a queue entry is sent via PNET to another node, the checkpoint with extended information is not sent to the other node.
2. If a queue entry is written to tape (by means of the POFFLOAD command), the checkpoint with extended information is not written to tape.

Deletion of a Checkpoint with Extended Information

The checkpoint information is deleted if any of the following is true:

- You have issued the request to delete checkpoint information (for details, see "Deleting Checkpoint Information" on page 70).
- The queue entry is deleted.
- Another checkpoint request is issued. Because each queue entry may have only one checkpoint, a second checkpoint request replaces any previously recorded checkpoint information. Note that a checkpoint request without extended checkpoint information clears any extended information previously recorded as well.

If a queue entry has been processed successfully and remains on the spool file, (usually, this means the disposition of the queue entry changes to L or H) the extended checkpoint information is *not* deleted but remains available.

Storage Requirements

The maximum length of a checkpoint with extended information is equal to the size of a data block minus 288 bytes. The size of a data block is specified in the DBLK operand of the VSE/POWER macro and must be a number from 1,000 to 65,024. The 288 bytes are reserved for VSE/POWER internal control information. Thus, the length of the checkpoint with extended information can be any number between 1 and 64,736 bytes (depending on the DBLK size).

Recording a Checkpoint with Extended Information

Extended checkpointing is invoked by the XPCC FUNC=SENDR macro instruction. It sends a checkpoint control record containing the extended checkpoint information to VSE/POWER. The checkpoint control record can be sent at any time while processing a queue entry via the GET service (unless you are in BROWSE mode).

To allow for recording of a checkpoint with extended information follow these steps:

1. Record the checkpoint with extended information the same way as the normal checkpoint described under "Requesting a Checkpoint" on page 88.
2. In addition, update the following fields within the checkpoint control record:
   a. PXCPFXIE in PXCPFLAG

b. Length of variable checkpoint with extended information plus length of fixed part of checkpoint control record in PXCPRLEN
c. Extended information starting at label PXCPSTXI.

After having successfully recorded the requested checkpoint, VSE/POWER returns a checkpoint-response control record. The extended information is not reflected within the checkpoint-response control record and the length field PXCRRLEN contains the length of the checkpoint-response control record without the extended information. PXCRFXIS is set within PXCRFLAG indicating that the extended checkpoint information has been saved.

For all checkpoint requests, regardless if extended checkpoint information has been specified or not, the spool-access support user receives a checkpoint-response control record which contains the queue entry number (PXCRQNUM). If the reply buffer of the spool-access support user is too short to contain a checkpoint-response control record, VSE/POWER sends a 'short' checkpoint-response control record, which means that the length of the checkpoint-response control record is equal to the length specified by the spool-access support user.

Retrieving a Checkpoint with Extended Information

Whenever a GET OPEN request has been issued, a verification SPL is passed back in the reply buffer. This SPL contains, for example, the following information about a checkpoint which has been requested during previous processing of this queue entry.

1. SPLDCCPY containing checkpoint copy number
2. SPLDCREC containing checkpoint record number
3. A bit SPLDFCKI within SPLDFLG indicating that extended checkpoint information exists
4. A bit SPLDFCKE within SPLDFLG indicating that extended checkpoint information exists, but is 'not available due to an I/O error'.
5. A 2 byte field SPLXCKIL containing the length of the checkpoint with extended information.
6. A 4 byte field SPLXQNUM containing the queue entry number (this number must be used if the checkpoint information should be deleted by using the CTL request 'delete checkpoint information', see “Deleting Checkpoint Information” on page 70).

If the user wants to restart on the checkpoint record and to retrieve the extended checkpoint information for printer setup, a 'retrieve extended checkpoint information' request must be indicated in the action byte of the user data in the XPCCB and passed to VSE/POWER with a null buffer (buffer with record length of zero). VSE/POWER, then, passes the extended checkpoint information to the user. The user may, then, continue with the restart control record.

Issuing a Retrieve Extended Checkpoint Information Request

You can retrieve a checkpoint with extended information at any time while processing a queue entry via the GET service, unless you are in BROWSE mode. Provide the following information within the XPCCB macro and issue an XPCC request with option FUNC=SENDR:

1. Set request PXUATCKR in PXUACT1.
2. Set PXUBTYP to zero.
3. Set up fields to send a null buffer (field IJBXBLN set to zero).

4. Set up fields for a reply buffer.

VSE/POWER returns the checkpoint with extended information within the checkpoint-response control record. The extended information starts at location PXCRSTXI. Additionally, PXCRFXIE within PXCRFLAG is set indicating that extended checkpoint information is returned.

Checking the Return Information

Errors During Recording

If during the recording of a checkpoint, the length of the extended checkpoint information is invalid, a return code X'08' together with the feedback code PXP08CKZ or PXP08CKL is set up in the user data field of the XPCCB and a null buffer is returned. If the checkpoint with extended information is too large (PXP08CKL), you may consider increasing the DBLK size (defined within the VSE/POWER generation macro) and perform a cold start.

Errors During Retrieving

If the spool-access support user tries to retrieve a checkpoint with extended information, but no extended checkpoint information has been previously saved, a return code X'04' together with a feedback code PXP04CKN is set up in the user data field of the XPCCB and a null buffer is returned.

Once extended checkpoint information has been successfully recorded, it indicates that the extended checkpoint information has been written to disk. When retrieving the extended checkpoint information, an I/O error may occur and the extended checkpoint information can no longer be read from the disk. Likewise, the VSE/POWER internal control information might have been destroyed. In both cases, the extended checkpoint information is no longer available for the spool-access support user. These cases are described in the following subsection.

If the spool-access support user issues a GET OPEN request, a verification SPL is returned to the user within the reply buffer. At this time, VSE/POWER tries to read the extended checkpoint information. If a 'retrieving error' occurs, the verification SPL contains both the indication 'extended checkpoint information exists' (SPLDFCKI) and 'extended checkpoint information is not available due to an I/O error' (SPLDFCKE). In this case, a value of zero is returned for the length of the extended checkpoint information.

Even when the verification SPL indicates that extended checkpoint information exists, a retrieving error might occur later, when retrieving the extended checkpoint information. In this case, a return code X'04' together with a feedback code PXP04CKE is set up in the user data field of the XPCCB and a null buffer is sent back.

Even if the recording of extended checkpoint information and its retrieval occurs during one GET service, a 'retrieving error' may occur. The same happens as described above: a return code X'04' together with a feedback code (PXP04CKE) is set up in the user data field of the XPCCB and a null buffer is sent back.

Requesting a Restart of the GET Spool Data

Your program can request VSE/POWER to restart retrieval at any point during GET data processing. It can request such a restart immediately after processing of the OPEN request is complete; it can, in fact, request a restart even after the end-of-data indication has occurred, but before it passes the end-service request.

To track the progress of an active job or output, a concurrent SAS GET BROWSE task may issue a "Restart to Active Record" request to position itself on the last record processed by the task keeping the queue entry active.

Table 27 shows a sequence diagram for a restart request. The diagram assumes that GET service processing has been opened successfully. Chapter 13, “Spool-Access Support Programming Example,” on page 271 includes a restart request at label GETB3.

Table 27. Restart of a GET Service Sequence

Coding in your application program	Comments
... ...	
WAIT IJBXSECB	Wait for the SENDR ECB to be posted. VSE/POWER returns a verification SPL to your program's reply buffer.
Check the reason code (in the XPCCB (byte IJBXREAS).	
Pick up and evaluate the verification SPL, if necessary.	
Restart request XPCC FUNC=SENDR,...	Your program's send buffer must contain a restart control record.
Check the return codes in register 15 and in the XPCCB (byte IJBXRETC).	
WAIT IJBXSECB	Wait for the SENDR ECB to be posted. VSE/POWER transfers data records to your program’s reply buffer, as many records as will fit.
Check the VSE/POWER return and feedback codes (in the XPCCB bytes PXPRETCD and PXPFBKCD, respectively).	
... ...	At this point, the coding sequence is the same for the retrieval of a complete queue entry.

To make a restart request, your program must:

1. Set byte PXUACT1 of the XPCCB to zero.
2. Set byte PXUBTYP of the XPCCB to the value equated to PXUBTCTL.

   This tells VSE/POWER that your program's send buffer contains a control record.

3. Set up a restart control record in your program's send buffer.

   By issuing a PWRSPL macro with TYPE=MAP, the assembler generates the restart control record DSECT labeled PXRSDSCT. In the record, set field PXRSOPT to:

   X'00' if the number in field PXRSRECN is a spool-record (card) number.

   X'20' if the number in field PXRSRECN is a page number (this option is ignored for RDR/PUN type entries).

   X'80' if the number in field PXRSRECN is a line number (this option is ignored for RDR type entries).

   If an output queue entry is being retrieved, you can specify a copy number in field PXRSCOPN of the control record. The number tells VSE/POWER that it is to restart retrieval at the specified record in the specified output copy. If you set the field to zero, VSE/POWER uses the current number-of-copies count. As a help in defining a restart point, VSE/POWER passes to your program the internal record count found in field RECLOGNO of the prefix of every retrieved record.

   This internal record number starts with 1 for the first record of the job/output and is incremented by 1 not only for normal data records but also for control records of printers and punches. The total number of records of each queue entry is shown in field 'QRNR' (of the internal queue record mapped by IPW$DQR) and in 'SPLDRCT' of the SPL. The total number of data records (lines/cards) is shown in field 'QRLC' of the queue record and in SPL field 'SPLDLCT'.

   RDR queue entries contain no control records. Therefore 'QRLC' and 'SPLDLCT' are 0 and 'QRNR' and 'SPLDRCT' show the total record count (which is also the data record count).

   Spool-Access Support programs retrieving only data records but no control records (PWRSPL OPT=CTLREC not set) will not receive the internal record number in consecutive order. There will be gaps when control records exist, since these are not passed to the program. For RDR queue entries, such gaps do not exist because they contain only data records.

   If your program re-accesses a previously retrieved and checkpointed queue entry, VSE/POWER returns the last recorded checkpoint information in the verification SPL as follows:

   - The number of the record last checkpointed, in field SPLDCREC.
   - The related copy number, if applicable, in field SPLDCCPY.

4. Issue an XPCC FUNC=SENDR request.

   The request passes to VSE/POWER the restart-control record set up by your program in its send buffer.

In response to a valid restart request, VSE/POWER repositions the retrieval pointer. VSE/POWER then continues processing by passing records to your reply buffer, starting with the record or line defined in the restart control record.

Note: A restart with a record number of zero will return an SPL as the first record of the data. This SPL is called an *inline SPL*, which reflects the Data Set Header Record.

Restarting to the Active Record During GET BROWSE

This request is suitable only if you have accessed an active queue entry (DISP=*) with GET BROWSE.

To set up a *restart to active record* request, your program must

1. Set byte PXUACT1 of the XPCCB to zero.
2. Set byte PXUBTYP of the XPCCB to the value equated to PXUBTCTL. This tells VSE/POWER that your program's send buffer contains a control record.
3. Set up a restart control record in your program's send buffer. By issuing a PWRSPL macro with TYPE=MAP, the assembler generates the restart control DSECT labeled PXRSDSCT. In the record, set field PXRSOPT to:
   - **X'10'** (PXRSOPAR) requesting 'Position on Active Record'

   Note that PXRSRECN (logical record number, where to restart) and PXRSRCPY (associated restart copy number) will then be ignored.
4. Issue a SENDR request.

In response to such a restart request, VSE/POWER will

- either reject the request with return/feedback code
  - PXP04NAT (no active task found on same system), if the queue entry is not active at all or if it is active on another system of a shared spooling complex.
  - PXP04ANS (active task not suitable), if there is an update task found on the same system as the browse task but the update task is not suitable for this request.
  - PXP04RIS (restart request with inconsistent specification), if PXRSOPOP (positioning on pages requested), PXRSOPOL (positioning on lines requested), or PXRSOPAE (positioning at end, if number too high) is set together with PXRSOPAR.
  - PXP04NRU (no restart to active request allowed for update task), if the requesting program is not a browser but has accessed the queue entry for update (normal GET-OPEN).
- or it will adjust the retrieval point to the last record (also 'active') processed by an update task that processes the same queue entry on the same VSE/POWER system as the GET BROWSE task. VSE/POWER then continues passing records to your reply buffer, starting with the 'active' data record.

  Note: The *last record processed* is the last record fetched by the VSE/POWER spool data management function and passed to the update task. For update tasks using buffered write to an external resource or program, this will in most cases be a few records ahead of the record written to the external resource or handled by the receiving program.

- or if the update task has just started and has not yet handled a data record, the retrieval point is adjusted to zero and the reply buffer starts with the SPL as the first record, followed by the records of the selected entry. In this case
  - PXPUSER field PXPBTYP shows normal data buffer (PXPBTNDB).
  - PXPUSER fields PXPLC12 and PXPLC34 contain zero (see "Identifying the Position after Restart to Active" on page 96 for the usage of PXPLC12 and PXPLC34).
  - RECPRFIX field RECTYPE shows SPL (RECTSPL) for the first record in the buffer.

  For example: When a LST task is waiting for forms requested by message

```
1Q40A ON cuu FORMS fno       NEEDED FOR jobname jobnumber
```

the LST task has not yet processed a data record. A browse task accessing the queue entry concurrently and requesting a *restart to active record* request, will receive such a reply buffer and must be prepared to handle it.

Identifying the Position after Restart to Active

Some programs like VSE/ICCF are interested only in data records and have built their own counting mechanism for them. For each record, they maintain its data record number as a consecutive number, starting with 1 for the first data record and ending with the number contained in field QRLC (of the internal queue record mapped by IPW$DQR) for LST/PUN queue entries or in field QRNR for RDR queue entries. Such programs want to be informed about the data record number of the first data record in the reply buffer of the 'Position on Active Record' restart request, in order to synchronize their own counting with VSE/POWER again.

Therefore, VSE/POWER returns this restart data record number (in addition to the internal record number RECLOGNO) to the application program by splitting the 4-byte record number into two 2-byte parts in fields PXPLC12 and PXPLC34, which belong to PXPUSER section described in PWRSPL.

Issuing Requests Concerning an OPTB

Issuing a Get-OPTB Request

Your program can request VSE/POWER to retrieve either all available OPTBs (output parameter text block) or a specific OPTB.

- OPTBs are contained in an output queue entry if the * $$ LST or * $$ PUN statement includes any user-defined keywords that have been defined in autostart DEFINE statements.
- OPTBs can also be passed to VSE/POWER as an appendage of the SPL (Spool Parameter List) at PUT OPEN time (see “Output Parameter Text Blocks (OPTBs)” on page 132).

You can send the Get-OPTB control record to VSE/POWER at any time while accessing an output queue entry (during GET data processing) or while spooling output data (PUT function). If OPTBs are present, the SPL contains a two-byte field indicating the total length of all OPTBs (see Figure 3 on page 132 and Figure 5 on page 134).

You can obtain the format of the GET-OPTB control record by issuing a PWRSPL macro with TYPE=MAP. The assembler generates the GET-OPTB control record DSECT labeled PXGODSCT. In the control record, pass the desired OPTB ID in field PXGOID.

If you specify an OPTB ID in the control record, VSE/POWER places only this particular OPTB into your program's reply buffer. If you do not specify an OPTB ID, VSE/POWER places *all* OPTBs into the reply buffer.

To obtain one or more OPTBs your program must:

1. Set up a Get-OPTB control record in your program's send buffer.
2. Set byte PXUACT1 of the XPCCB to zero.
3. Set byte PXUBTYP of the XPCCB to the value equated to PXUBTCTL.

This tells VSE/POWER that your program's send buffer contains a control record.

4. Issue an XPCC FUNC=SENDR request.

   The request passes to VSE/POWER the Get-OPTB control record set up by your program in its send buffer.

Issuing a Modify-OPTB Request

Your program can request VSE/POWER to modify an existing OPTB. Via the Modify-OPTB control record you can update (overwrite) any OPTB with a new one, which must have the same length as the old OPTB. You can send the Modify-OPTB control record to VSE/POWER at any time while accessing an output queue entry (during GET data processing) or while spooling output data (PUT function), but not when you are in browse mode.

You can obtain the format of the Modify-OPTB control record by issuing a PWRSPL macro with TYPE=MAP. The assembler generates the Modify-OPTB control record DSECT labeled PXMODSCT. In the control record, pass the OPTB to be modified starting at field PXMOOPTB.

To modify one or more OPTBs your program must:

1. Set up a Modify-OPTB control record in your program's send buffer.
2. Set byte PXUACT1 of the XPCCB to zero.
3. Set byte PXUBTYP of the XPCCB to the value equated to PXUBTCTL.

   This tells VSE/POWER that your program's send buffer contains a control record.
4. Issue an XPCC FUNC=SENDR request.

   The request passes to VSE/POWER the Modify-OPTB control record set up by your program in its send buffer.

Checking the Return Information for GET-Service Requests

For the return information to be checked by your program after an XPCC request, refer to “XPCC” on page 212.

For every GET-service request, your program should check return information supplied by VSE/POWER. Provide for this checking after your program's SENDR ECB has been posted.

Table 28 on page 98 lists the return and feedback codes that VSE/POWER may supply when it processes a GET-service related request. The list is ordered in ascending order by code values; it relates the codes to the applicable request types; it gives the names that are equated to the feedback codes.

A complete list of the VSE/POWER return and feedback codes is given in the DSECT PXPUSER, which the assembler generates for a PWRSPL TYPE=MAP macro. You find the return codes at label PXPRETCD and the feedback codes at label PXPFBKCD, and the feedback-2 codes at label PXPFBKC2.

Table 28. Return and Feedback Codes (PXPRETCD/PXPFBKCD) for GET-Service Requests (Part 1)

			Request Type						
Mnemonic PXPFBKCD	**Return Code**	**Fdbk Code**	**GET-OPEN**	**GET-OPEN BROWSE**	**GET Data**	**Check point**	**Re-start**	**GET OPTB**	**Mod. OPTB**
PXP00OK	00	00	X	X	X	X	X	X	X
PXP00EOD		01			X		X		
PXP04NOF [2]	04	01	X	X					
PXP04JOP		02	X	X					
PXP04BSY		03	X	X					
PXP04NDS		04	X						
PXP04RER		06					X		
PXP04CER		07				X			
PXP04SOA		09	X	X					
PXP04BER		0A							X
PXP04ONF		11						X	X
PXP04CKN		13				X			
PXP04CKE		14				X			
PXP04SAC		17	X	X					
PXP04NAT		18					X		
PXP04ANS		19					X		
PXP04RIS		1A					X		
PXP04NRU		1B					X		
PXP08SPL	08	01	X	X					
PXP08REQ		02	X	X					
PXP08JNM		05	X	X					
PXP08QID		06	X	X					
PXP08CLS		07	X	X					
PXP08PWD		08	X	X					
PXP08UID		09	X	X					
PXP08BTS		1A			X	X	X	X	
PXP08IAB		1C			X				
PXP08ICR		1D				X	X	X	X
PXP08CON		22	X	X	X	X	X		
PXP08IBT		24	X	X		X	X		
PXP08ROS		25			X				
PXP08SOS		26	X	X					
PXP08BOS		27				X	X		
PXP08FB1		2B	X	X					
PXP08JNO		31	X	X					
PXP08JSF		32	X	X					
PXP08IRR		38						X	X
PXP08IOP		39							X
PXP08OLM		3A							X
PXP08IDH		3D						X	X
PXP08CKZ		42				X			
PXP08CKL		43				X			
PXP08IQN [1]		44	X						
PXP0CINS	0C	01	X	X	X	X	X		
PXP0CIXF		02	X	X	X	X	X		
PXP0CIOE		07	X	X	X	X	X		
PXP10PSP	10	05	X	X	X	X	X		
PXP10SIE		06	X	X	X	X	X		
PXP10MST		07	X	X					

Table 28. Return and Feedback Codes (PXPRETCD/PXPFBKCD) for GET-Service Requests (Part 1) (continued)

			Request Type						
Mnemonic PXPFBKCD	**Return Code**	**Fdbk Code**	**GET-OPEN**	**GET-OPEN BROWSE**	**GET Data**	**Check point**	**Re-start**	**GET OPTB**	**Mod. OPTB**

Note:
1. This feedback code appears only for direct GET Service requests.
2. If returned for a direct GET Service request, check also PXPFBKC2 of Table 30 on page 100 for detailed reason.

Table 29. Return and Feedback Codes (PXPRETCD/PXPFBKCD) for GET-Service Requests (Part 2)

			Request Type				
Mnemonic PXPFBKCD	**Return Code**	**Fdbk Code**	**PURGE**	**CLOSE**	**QUIT**	**QUIT LOCK**	**FLUSH HOLD[1]**
PXP00OK	00	00	X	X	X	X	X
PXP00EOD		01					
PXP04NOF	04	01					
PXP04JOP		02					
PXP04BSY		03					
PXP04NDS		04					
PXP04RER		06					
PXP04CER		07					
PXP04SOA		09					
PXP04BER		0A	X	X	X	X	X
PXP08SPL	08	01					
PXP08REQ		02					
PXP08JNM		05					
PXP08QID		06					
PXP08CLS		07					
PXP08PWD		08					
PXP08UID		09					
PXP08BTS		1A					
PXP08IAB		1C	X	X	X	X	X
PXP08ICR		1D					
PXP08CON		22	X	X	X	X	X
PXP08IBT		24					
PXP08ROS		25	X	X	X	X	X
PXP08SOS		26					
PXP08BOS		27					
PXP08RPH		28					X
PXP08FB1		2B					
PXP08JSF		32					
PXP0CINS	0C	01	X	X	X	X	X
PXP0CIXF		02	X	X	X	X	X
PXP0CIOE		07	X	X	X	X	X
PXP10PSP	10	05	X	X	X	X	X
PXP10SIE		06	X	X	X	X	X

Note: [1] The FLUSH HOLD function is part of the external device support.

Table 30. Feedback-2 Codes (PXPFBKC2) for Direct GET-Service Requests

			Request Type
Mnemonic PXPFBKCD/PXPC2BKC2	Return/Feedback Code	Feedback-2 Code	GET-OPEN
PXP04NOF/PXPC2BAD	04/01	08	X
PXPC2FRE		09	X
PXPC2MQU		0A	X
PXPC2MJM		0B	X
PXPC2MJB		0C	X
PXPC2MJS		14	X
PXPC2MCL		15	X
PXPC2MSY		16	X
PXPC2MFU		17	X
PXPC2MFT		18	X
PXPC2SAC		19	X
PXPC2INC		1A	X
PXPC2DEL		1B	X
PXPC2NVT		1C	X
PXPC2EMP		1D	X
PXPC2QCL		1E	X
PXPC2QCP		1F	X
PXPC2QCR		20	X
PXPC2QCX		21	X
Note: For a detailed explanation of the PXPFBKC2 mnemonics, refer to "Spool-Access Support Parameter List (PWRSPL DSECT)" on page 231.			

Handling an Abnormal-End Condition During GET

- For an abnormal end of your program or of the VSE/POWER service task:

  If the output being retrieved by the GET service has been created via the PUT service with the 'protect' option on, the queue entry is placed into the non-dispatchable queue with disposition Y.

  A queue entry is protected when the SPL field SPLDMOHP is set on to signal: 'Hold when print/punch fails' (see Table 36 on page 118).
- For an abnormal end of VSE/POWER or of the z/VSE system, the same applies as described above.

A queue entry with disposition Y is not automatically processed by the various VSE/POWER tasks. Your program can make use of the CTL service to

- Get a display of all queue entries that have a disposition of Y by entering the PDISPLAY ALL,CDISP=Y command, and
- Alter this disposition for a queue entry to make it eligible for processing again. To reset disposition Y of a queue entry to its original one, use the PALTER queue,jobname,DISP=* command.

For further information on disposition, see *VSE/POWER Administration and Operation*, SC34-2625.

Accessing the Transmit (XMT) Queue

Using the GET Service

The GET service ("Introduction to the GET Service" on page 75) of the spool-access support does not offer access to entries residing in the XMT-queue (see QUEUE= parameter of the PWRSPL macro in the topic "Format 3: Generating a DSECT" on page 219). However, the following sequence of access requests may be used to first transfer an XMT-queue entry to one of the local queues, then to GET-access the entry, and finally to return the entry to the XMT-queue:

1. Save disposition, node-destination, class and type (L=list, P=punch, J=job) of the XMT queue entry as obtained from a queue display.
2. Issue a spool-access support CTL request to alter the node to LOCAL and the disposition to H or L; according to L/P/J-type the entry is added to the LST/PUN/RDR queue - non-dispatchable, so that no local task may gain access to the entry.
3. Use GET BROWSE to access the non-dispatchable local entry in its corresponding queue while making use of the saved class.
4. Issue a CTL request to lift the entry back to the XMT-queue by altering its node destination and its disposition back to the original values.

Using the Direct GET Service

This support ("Direct Queue Entry GET Access to the RDR/LST/PUN/XMT Queues" on page 79) provides access to entries residing in the XMT queue. For details, see "Special Considerations for Access to the XMT Queue" on page 81.

Chapter 9. PUT - Submitting a Job, a Job Stream, or Output

Your program initiates PUT-service processing whenever data is to be submitted to VSE/POWER for inclusion in one of its queues. Jobs, including the associated input data, are submitted for inclusion in the RDR or XMT queue, whichever applies. Output data is submitted for inclusion in an output queue (LST, PUN, or XMT).

Submission for Inclusion in the XMT Queue

To submit a job for processing at another node (of your computer system's network), specify this in the * $$ JOB statement for the job.

To submit output data for transmission to another node, give the target node's name and the applicable user ID in the SPL fields SPLDTNN and SPLDTUID, respectively.

In the SPL macro, you specify QUEUE=RDR for job input; you specify QUEUE=LST for list output and QUEUE=PUN for punch output.

Data Format

The format of the data to be spooled is always the same. Every record must be preceded by the following eight-byte prefix. You get a DSECT of this prefix, labeled RECPRFIX, by issuing a PWRSPL macro with TYPE=MAP.

```
   Bytes                        Meaning
--------------------------------------------------------------------
   0       Carriage control character, if any
   1       Record type:
             X'00' = A normal data record
             X'06' = A CPDS (composed page data stream) record
  2-3      Length of a logical record (in binary notation)
  4-7      Reserved
```

Data Lengths

For type X'00' (normal) records, the minimum, maximum and default lengths are:

Table 31. Data Length for PUT Service

	Job Data	**Output LST Data**	**Output PUN Data**
Min. Record Length	80	1	80
Max. Record Length	128	32 KB minus 8	32 KB minus 8
Default (see [1])	80	512	80

[1] The default is assumed by VSE/POWER if your program does not define a data length in field SPLDLREC of the SPL.

If an output-spool record includes trailing blanks, your program can truncate these blanks prior to passing the record to VSE/POWER. This makes better use of send buffer space.

If a record to be passed is longer than the specified maximum length, VSE/POWER truncates the record and informs your program by a feedback code; VSE/POWER spools the truncated record as well as the remaining records in the passed data buffer. For a passed record shorter than the specified maximum length, VSE/POWER:

- Expands this record by padding it with blanks at the end if a job is submitted. When the record is stored on disk, trailing blanks can be truncated according to the * $$ JOB BTRNC=YES|NO setting.
- Spools the record as presented if output is submitted. When the record is stored on disk, trailing blanks will be truncated unless SPLGO2BT has been specified in the SPL.

Size of Buffers

Your program must define the sizes of your send and reply buffers.

The Send Buffer

The buffer must be large enough to hold your program's SPL when the processing of the desired service is initiated. It must be large enough to hold the longest record (including the eight-byte prefix) that is to be passed to VSE/POWER.

To pass data to VSE/POWER for spooling, your buffer should have a length equal to the sum of the lengths of the data records (including the record prefix) that your program is to submit at a time. This may be just one record or a number of records. A zero data length field in a record prefix is an end-of-buffer indication for VSE/POWER. If set erroneously, it may lead to unexpected RC/FB codes returned by VSE/POWER, such as 00/02=PXP00NJB (job not on job boundary).

You use the BUFFER operand of the XPCC macro or the XPCCB macro to define the buffer.

The Reply Buffer

The buffer must be large enough to hold a verification SPL passed to your program by VSE/POWER. You use the REPAREA operand of the XPCCB macro to define the buffer.

Retrieval of Messages

VSE/POWER collects all job- or output-submission error or warning messages that would normally go to the system console. They enable your program to determine whether the job- or output-spool operation was completed successfully; they inform your program about possible errors and unusual conditions, if any.

Note: Job event and output generation messages cannot be retrieved. For retrieval of such messages, see Chapter 10, “GCM - Retrieving Job Event and Output Generation Messages, Inquiring eXtended Event Messages,” on page 139.

Following your PUT-CLOSE request, VSE/POWER sets info byte PXPINFO of your program's XPCCB to the value equated to PXPIMSG if any messages have been queued. Your program can request these messages to be passed by VSE/POWER (for returning submission error and warning messages, see “Issuing a RETURN-MESSAGE Request” on page 113).

If your program does not request the messages to be returned, VSE/POWER discards them on receipt of the next service request (CTL, GET, or PUT specified in FUNC=code of the PWRSPL macro).

Messages returned by VSE/POWER can be up to 132 bytes long; they are preceded by an eight-byte header with the following contents:

```
  Bytes            Contents/Meaning
----------------------------------------------------
    0         X'00'  Set by VSE/POWER
    1         X'02'  Set by VSE/POWER
   2-3        Length of message (in binary)
   4-7        Reserved
```

A reply buffer of 700 bytes, for example, can hold up to five messages of maximum length.

Submitting a Job or a Job Stream

This PUT service spools the submitted records as a queue entry in the RDR (XMT) queue. In your program, you can issue job-related PUT-service requests as follows:

- A *PUT-OPEN* request to start the spooling of one or more jobs – For details, see "Starting a PUT Service for a Job or a Jobstream" on page 106.
- One or more *PUT-SPOOL data* requests to have VSE/POWER spool the submitted job(s) – For details, see "Issuing a PUT-SPOOL-Data Request" on page 111.
- A *PUT-CLOSE* request to indicate that the submission of job data is finished and that the submitted job data is to be included in VSE/POWER's input queues – For details, see "Issuing a PUT-CLOSE-Service Request" on page 111.
- A *PUT-QUIT* request to indicate that no further data is to be submitted for the currently processed job and that the job should not be included in VSE/POWER's input queues – For details, see "Ending the PUT Service for Jobs" on page 112.

When your program submits job records, VSE/POWER does not insert any JECL statements. In other words, JECL statements required by VSE/POWER are to be supplied by your program preceding the job records. If a VSE/POWER * $$ JOB statement is not provided, then the user may supply the job name and job user information via the // JOB statement, and some other optional information via the PWRSPL (see Table 33 on page 108).

If there is a user-written JOBEXIT routine for local input, VSE/POWER passes to the routine the z/VSE job-control and JECL statements of the submitted jobs.

With one PUT-service request, your program can submit just one job or a job stream consisting of two or more jobs. However, submission of just one job per service request is the preferred method; it makes evaluation of returned messages easier. VSE/POWER does not return messages to your program until the end of data has been reached. As a result, a clear distinction which message belongs to which job is difficult if you submit several jobs following a PUT-service OPEN request.

After having processed one of the following, VSE/POWER returns to your program a *verification SPL*:

- Your PUT-OPEN request.
- Your PUT-CLOSE request.

- A PUT-SPOOL data request for a buffer containing two or more jobs if a short-on-account-space error occurs.

Besides the data supplied by your program in its SPL, a verification SPL contains:

- Default values for fields not set in your program's SPL.
- Statistics such as the total number of records spooled for your job, jobnumber, jobsuffix, and queue entry number, if the verification SPL is passed by VSE/POWER following a CLOSE request.

Coding Sequence for a PUT-JOB Request

Refer to Table 32, a coding sequence diagram for the submission of a job to an input queue. Table 32 shows the kind of coding you have to supply in your program and in what sequence this coding is to be. Chapter 13, “Spool-Access Support Programming Example,” on page 271 includes a PUT-job service request at label PUTA1.

Format of the Spool Job Records

Every job record that is to be spooled by VSE/POWER must have an 8-byte prefix as shown in Table 24 on page 77. The record prefix must be updated for the following fields (refer to “Spool-Access Support Parameter List (PWRSPL DSECT)” on page 231):

RECTYPE
record type, set to RECTNORM=X'00', normal data record

RECLNGTH
length of the subsequent logical record

The coding sequence of label FILLBUF in Chapter 13, “Spool-Access Support Programming Example,” on page 271 shows how to set up the record prefix for job records.

Starting a PUT Service for a Job or a Jobstream

To open a PUT service for the submission of a job, VSE/POWER requires:

- Byte PXUBTYP of the XPCCB to be set to the value equated to PXUBTSPL. This indicates to VSE/POWER that the send buffer contains an SPL.
- An SPL as set up by a PWRSPL macro with TYPE=GEN or updated by a PWRSPL macro with TYPE=UPD.

  VSE/POWER requires control data to be passed in your program's SPL in addition to that specified in the JECL statements for the job. Table 33 on page 108 lists the applicable SPL fields. For the lengths and data types of these fields, see the SPL DSECT which you get by issuing a PWRSPL macro with TYPE=MAP. Examine the fields of the SPL DSECT (at label SPLDS) and decide which of the SPL fields your program should set or change prior to the request.
- A reply buffer to which VSE/POWER passes the verification SPL.

Table 32. PUT Service, Job Submission Sequence

Step	Coding in your application program	Comments
	... ...	

Table 32. PUT Service, Job Submission Sequence (continued)

Step	Coding in your application program	Comments
1	Open the request XPCC FUNC=SENDR,...	Your program's send buffer must contain an SPL-generated (or updated) for processing a PUT-job service request.
2	Check the return codes in register 15 and in the XPCCB (byte IJBXRETC).	
3	WAIT IJBXSECB	Wait for the SENDR ECB to be posted. VSE/POWER passes a verification SPL to your program's reply buffer.
4	Check the reason code (in the XPCCB byte IJBXREAS).	
5	Check the VSE/POWER return and feedback codes (in the XPCCB bytes PXPRETCD and PXPFBKCD, respectively).	
6	Pick up and evaluate the verification SPL, if necessary.	
7	PUT-Data Request XPCC FUNC=SENDR,...	Your program's send buffer must contain the records which VSE/POWER is to spool.
8	Check the return codes in register 15 and in the XPCCB (byte IJBXRETC).	
9	WAIT IJBXSECB	Wait for the SENDR ECB to be posted. It indicates to your program that VSE/POWER has finished processing the records in the send buffer.
10	Check the reason code (in the XPCCB byte IJBXREAS) Check the VSE/POWER return and feedback codes (this is the same as above in steps 4 and 5). Either Fill your buffer with records for the next request and **return to Step 7**. Or proceed to the next step.	Loop until all records for the queue entry have been passed.
11	CLOSE request XPCC FUNC=SENDR,...	Your program can make this request with data in its send buffer or with a null buffer being passed to VSE/POWER.
12	Check the return codes in register 15 and in the XPCCB (byte IJBXRETC).	
13	WAIT IJBXSECB	Wait for the SENDR ECB to be posted. VSE/POWER passed a verification SPL to your program's reply buffer.

Table 32. PUT Service, Job Submission Sequence (continued)

Step	Coding in your application program	Comments
14	Check the reason code (in the XPCCB byte IJBXREAS). Check the VSE/POWER return and feedback codes (this is the same as above in steps 4 and 5). Pick up and evaluate the verification SPL, if necessary.	
15	End of Service	

Table 33. SPL Fields Applicable to a PUT-Job Service Request

Name of Field		Purpose/Contents
SPLGQI	M	Queue ID [1]
SPLGUS	M	User ID [1]
SPLGOPT2	O	Option byte (set SPLGO2BT, if trailing blanks of records should not be truncated during spooling)
SPLDPRGN	O	Programmer name [2]
SPLDROOM	O	Room number [2]
SPLDDEPT	O	Department number [2]
SPLDBLDG	O	Building number [2]
SPLDLREC	O	Maximum record length
SPLGFB1	O	Set SPLGF1QM if you want to store the job completion message in a message queue after the job finished processing. [3] Set SPLGF1QQ if you want to store all job event (completion and generation) messages in a message queue. [3] Set SPLGF1QP if you want to store the job completion and output generation messages (after output is created and ready for processing) in a message queue. [3] Set SPLGF1QO if you want to store the output generation message in a message queue.[3] Set SPLGF1QX if you want to store the job event (completion and generation) and output generation messages in a message queue. [3]
SPLXSID	O	z/VSE Security user id [2]
SPLXSPW	O	z/VSE Security password [2]

[1]Normally defined in the PWRSPL macro along with other spool-control values.

[2]An * $$ JOB specification overrides these operands.

[3]For retrieval of the messages, see Chapter 10, "GCM - Retrieving Job Event and Output Generation Messages, Inquiring eXtended Event Messages," on page 139.

Legend: M = Mandatory; O = Optional

You might have to supply additional spool-control values for the above request (for example job name) using an * $$ JOB statement. Submit this statement as the first step in the jobstream.

Enabling Retrieval of Job Event and Output Generation Messages

With the PUT-OPEN request it is also possible to request queueing of **event** messages to a specific queue identified by the XPCC-applid and Spool-Access user-id (SPLGUS) of the job submitter. Event messages can be the following:

- **job completion message (JCM)** 1Q5DI is implemented to check for successful execution of a submitted job (for layout and contents, refer to "Spool-Access Support Parameter List (PWRSPL DSECT)" on page 231 for additional clarification).
- **job generation message (JGM)** 1Q5HI is implemented to identify when a submitted job creates another job by means of a * $$ PUN statement with the DISP=I operand (for layout and contents, refer to "Spool-Access Support Parameter List (PWRSPL DSECT)" on page 231 for additional clarification).
- **output generation message (OGM)** 1Q5RI is implemented to check for successful creation of LST and PUN outputs. This message is issued when the output has been created and is ready for processing (for layout and contents, refer to "Spool-Access Support Parameter List (PWRSPL DSECT)" on page 231 for additional clarification). If a job produces segmented output (for example, due to RBS= operand in the * $$ LST JECL statement, or PSEGMENT operator command), then an output generation message is issued for each segment. This message is also issued for every duplicate, which is generated due to * $$ LSTDUP or * $$ PUNDUP JECL statement and due to DUP=YES operand passed in * $$ LST or * $$ PUN statement within the JECL area of the IPWSEGM macro. Output generation message is not generated for output entry spooled to tape, that is for entry spooled as a result of the TADDR=cuu option, or TDISP=T one, or both in the * $$ LST or * $$ PUN statement.

These messages can be retrieved by the job-submitting application or by any other application as described in Chapter 10, "GCM - Retrieving Job Event and Output Generation Messages, Inquiring eXtended Event Messages," on page 139.

Requesting Job Event and Output Generation Messages

To request job event messages or output generation messages, set up byte SPLGFB1 in the PUT-OPEN SPL with one of the following specifications:

1. SPLGF1QM - to request creation of a subset for job event messages, namely job completion messages.
2. SPLGF1QQ - to request creation of all job event messages. This means that both, job completion and job generation messages, will be created.
3. SPLGF1QP – to request creation of both job completion and output generation messages.
4. SPLGF1QO – to request creation of output generation messages only.
5. SPLGF1QX – to request creation of job event and output generation messages. This means that all possible messages will be created: job generation, job completion, and output generation messages.

In response to specification of the above listed options, VSE/POWER can inform your program by setting the bytes in PXPRETCD/PXPFBKCD in the cross-partition control block as follows:

PXPRCOK/PXP00OK (X'00'/X'00')
: Your job has been successfully submitted to VSE/POWER. No error occurred.

PXPRCOK/PXP00NCM (X'00'/X'07')
No message queue exists to which job event and output generation messages can be queued, because the message queue size has been set JCMQ=0 during VSE/POWER startup.

PXPRCOK/PXP00LCM (X'00'/X'08')
The space capacity of the message queue is nearly exhausted. Space remaining for only 2 to 5 messages in the queue.

PXPRCOK/PXP00OCM (X'00'/X'09')
The space capacity limit of the message queue is reached. Space remaining for possibly 1 message in the queue.

With all these return and feedback combinations submitted the job is accepted. You also have to prepare for other return and feedback codes, which VSE/POWER may return in response to a PUT-OPEN request; refer to Table 35 on page 116.

Jobs Generated with DISP=I

A job which has been generated as a result of DISP=I on the * $$ PUN statement will subsequently inherit the 'queue event message' characteristic of the parent job. That means, one or more fixed format job event, output generation messages, or both are also created for the child job, and can be retrieved in the same manner as the event message for the parent job.

Additional Job Event and Output Generation Message Options

When you want to enable creation of job event messages, output generation messages, or both of them, you can use additional options, which are dependent on your environment. These specified options provide additional information to your job and event messages being produced. However, the messages will only be effective if options SPLGF1QM/QQ/QP/QO/QX are specified. In addition, these options are also passed to jobs generated with an * $$ PUN DISP=I statement.

The following options exist:

1. SPLGOPT2

   For PUT service, you can only specify the option SPLGO2OJ in this option byte.

   This option is useful when the job is transmitted to another node and executed there. If the option is omitted, messages will contain the job number that the job receives at the final execution node. However, if SPLGO2OJ is set, the messages will also contain the job number that is obtained at the node where it was initially submitted. This job number is sent to your application within the verification SPL but only after the PUT CLOSE request has been successfully processed. Later, when retrieving the job event messages by the GCM service, specify the same SPLGO2OJ. VSE/POWER will interpret the SPL job number specification SPLGJN to your GCM request as the original job number. That is, it compares SPLGJN with the field JCMFONUM of JCM, field JGMF1NUM of JGM, and field OGMFONUM of OGM (instead of fields JCMFNUM, JGMFNUM, and OGMFNUM).
2. SPLXPRIV

   You can specify here any user private data. The data is not checked for any range of values, nor is it modified by the job. The data will finally be reflected in the resulting event message in the field JCMFPRIV for job completion event, in the field JGMFPRIV for job generation event, and in the field OGMFPRIV for output generation event.
3. SPLXOB1

This option byte is used to specify at job submission time the message queuing destination by setting the following options:

SPLXO1CQ
: adds messages only to the common message queue, which is defined by XPCC appl ID and artificial 8 bytes X'FF...FF' user ID.

SPLXO1DQ
: adds messages to both user and common message queues, which are defined by XPCC appl ID | SPLGUS user ID and XPCC appl ID | X'FF...FF' user ID. At first, a message is placed into the user message queue, and then into the common one.

SPLXO1CQ and SPLO1DQ not specified both
: adds messages only to the user message queue.

Use the common queue to collect all messages produced by jobs submitted under the same applid but with different userid, and to limit message retrieval to only one application program (that is to a single applid). It is possible to retrieve only those messages that were queued under the same applid that is used for retrieval.

It is likely that a common queue has to accommodate more messages than a single userid queue. Therefore, the capacity of a common queue has always the eightfold value of a single userid queue (defined by SET JCMQ=nnn). Several common message queues can exist.

Issuing a PUT-SPOOL-Data Request

After VSE/POWER has passed the verification SPL, your program must issue one or more spool-data requests, every one after the preceding one has been completed. To do this, provide that your program:

1. fills its send buffer with records to be spooled by VSE/POWER,
2. sets byte PXUBTYP of the XPCCB to the value equated to PXUBTNDB (This indicates that your program's send buffer contains records to be spooled.)
3. sets PXUACT1=0.
4. issues an XPCC FUNC=SENDR request.

VSE/POWER spools the records contained in your send buffer, except when an error condition is encountered. VSE/POWER indicates successful completion (or error, if any) to your program by way of return and feedback codes (PXPINFO).

Issuing a PUT-CLOSE-Service Request

A CLOSE request causes the data submitted up to this point to be placed into the RDR (XMIT) queue as a complete queue entry.

In your program, you can issue a CLOSE request either:

1. Together with passing the last buffer of spool records for the queue entry being submitted, or
2. Separately after your program has passed this last buffer.

For either case, set byte PXUACT1 of the XPCCB to the value equated to PXUATEOD before you issue the requesting XPCC macro. For case 1, this is all you have to do.

For case 2, a separate CLOSE request following the transfer of the last buffer, VSE/POWER requires that your program:

1. Sets byte PXUBTYP of the XPCCB to zero.
2. Sets up a null buffer (by setting field IJBXBLN to zero).
3. Issues an XPCC FUNC=SENDR request.

The coding sequence at label PUTA3 in Chapter 13, “Spool-Access Support Programming Example,” on page 271 shows how to issue a CLOSE request together with the last buffer of data records.

When it receives a CLOSE request, VSE/POWER expects the last record in the last buffer of spool records to be a valid job-end statement. If a valid job-end statement is not supplied, VSE/POWER automatically adds this statement and queues a message about this for your program.

When all records of your job are queued, VSE/POWER returns a verification SPL to your program's reply buffer. This SPL contains descriptive job information such as VSE/POWER assigned default values, the job name and number, and the queue entry number. However, if your program submits two or more jobs before it passes a CLOSE request, then the verification SPL reflects the characteristics of only the last job.

Ending the PUT Service for Jobs

In your program, you may have to provide for a quit-type end of service processing; that is, end of the opened processing without any data to be queued by VSE/POWER.

Your program can issue a QUIT request any time after an individual PUT-service request is complete (which is indicated by a posting of field IJBXSECB of the XPCCB). A QUIT request causes VSE/POWER to purge the queue entry that is being built. In case of a multijob submission, a job previously queued by VSE/POWER during the same PUT service remains unaffected.

Your program should check the QUIT-request return and feedback codes for successful completion of the request. This ensures that the communication path to VSE/POWER is free again to open another service request.

If additional jobs are to be submitted for spooling, your program must reopen the PUT service by issuing an XPCC macro that passes a suitable SPL.

To issue a QUIT request in your program:

1. Set byte PXUBTYP of the XPCCB to zero.
2. Set byte PXUACT1 of your XPCCB to the value equated to PXUATABR.
3. Issue an XPCC FUNC=SENDR request passing a null buffer, that is, a buffer with a length of zero (IJBXBLN set to zero).

This is the same as a QUIT request for a GET service. Chapter 13, “Spool-Access Support Programming Example,” on page 271 includes a coding sequence for a QUIT request at label GQUIT.

Issuing a RETURN-MESSAGE Request

You find a discussion of message retrieval under "Retrieval of Messages" on page 104.

VSE/POWER makes the submission error and warning messages generated during PUT service processing available on RETURN-MESSAGE request, and signals about such messages with the PXPIMSG flag within XPCC user information byte PXPINFO. Your program can pick them up in the defined reply buffer, one message behind the other.

Note: The RETURN-MESSAGE request does not retrieve job event or output generation messages. For retrieval of such messages, see Chapter 10, "GCM - Retrieving Job Event and Output Generation Messages, Inquiring eXtended Event Messages," on page 139.

If all messages fit into the reply buffer, VSE/POWER indicates this by the return- and feedback-code combination PXPRCOK and PXP00EOD. If additional messages are waiting to be transferred, VSE/POWER passes to your program a return- and feedback-code combination of PXPRCOK and PXP00OK. In that case, your program should issue another RETURN-MESSAGE request.

VSE/POWER deletes messages queued but not yet transmitted if your program does one of the following:

- Issues another, different open-service request passing a new SPL
- Issues a QUIT request
- Ends communication via the currently used path

Table 34, a coding sequence diagram, shows the kind of coding you have to supply in your program and in what sequence this coding is to be. Table 34 assumes that PUT-data requests have been serviced by VSE/POWER for the complete queue entry.

Chapter 13, "Spool-Access Support Programming Example," on page 271 includes a RETURN-MESSAGE request at label PUTA4. This coding sequence gets control if VSE/POWER passed XPCCB-user data with byte PXPINFO containing the value equated to PXPIMSG.

Table 34. Retrieve Messages after a PUT-Job Service Sequence

Step	Coding in your application program	Comments
	... ...	
1	CLOSE request XPCC FUNC=SENDR,...	Your program issues a CLOSE request when all records of a job have been submitted.
2	Check the return codes in register 15 and in the XPCCB (byte IJBXRETC).	
3	WAIT IJBXSECB	The WAIT required in your program to ensure that VSE/POWER has finished the necessary CLOSE processing. VSE/POWER returns a verification SPL.

Table 34. Retrieve Messages after a PUT-Job Service Sequence (continued)

Step	Coding in your application program	Comments
4	Check the reason code in the XPCCB. Check the VSE/POWER return and feedback codes. Pick up and evaluate the verification SPL, if necessary.	
5	Check user-information byte for queued messages. If no messages have been queued, **go to step 9** for ending the service processing. Else proceed.	VSE/POWER indicates the available of messages with the PXPIMSG flag.
6	Return-message request XPCC FUNC=SENDR,...	No SPL need be transferred for this request; your program must set a request code PUT in the XPCCB.
7	WAIT IJBXSECB	Wait for the SENDR ECB to be posted.
8	Check the XPCC reason code and VSE/POWER return and feedback codes (this is the same as above in step 4). If more messages are to be transferred by VSE/POWER, **return to Step 6**. Else proceed.	Loop until VSE/POWER returns the feedback code PXP00EOD.
9	End of Service	

To have VSE/POWER pass messages, your program must:
1. Set byte PXUBTYP of the XPCCB to zero.
2. Set byte PXUACT1 of the XPCCB to the value equated to PXUATRMR.
3. Issue an XPCC FUNC=SENDR request passing a null buffer, that is, a buffer with a length of zero (IJBXBLN set to zero). The coding sequence at label PUTA4 in Chapter 13, "Spool-Access Support Programming Example," on page 271 shows how to set up a null buffer.

Checking the Return Information for a PUT-Job Service Request

For the return information to be checked by your program after an XPCC request, refer to "XPCC" on page 212.

For every PUT-job service request, your program should also check the return information supplied by VSE/POWER. Provide for this checking after your program's SENDR ECB has been posted.

Table 35 on page 116 lists the return and feedback codes that VSE/POWER may supply when it processes a PUT-service related request for job submission. The list is in ascending order by code values. It relates the codes to the applicable request types and gives the names that are equated to the feedback codes.

A complete list of the VSE/POWER return and feedback codes is given in the DSECT PXPUSER, which the assembler generates for a PWRSPL TYPE=MAP macro. You find the return codes at label PXPRETCD and the feedback codes at label PXPFBKCD.

For more information on the subject, see Chapter 14, “Return and Feedback Codes and Their Meanings,” on page 297.

Table 35. Return and Feedback Codes for PUT-Job Service Requests

			Request Type				
Mnemonic	**Return Code**	**Feedback Code**	**PUT Open**	**PUT Data**	**CLOSE**	**QUIT**	**Get Message**
PXP00OK	00	00	X	X	X	X	X
PXP00EOD		01					X
PXP00NJB		02			X		
PXP00NRS		03			X	X	
PXP00RTR		04		X	X		
PXP00ZBF		05		X			
PXP00NCM		07	X				
PXP00LCM		08	X				
PXP00OCM		09	X				
PXP04SOD	04	08		X	X		
PXP04SOA		09	X				
PXP08SPL	08	01	X				
PXP08REQ		02	X				
PXP08QID		06	X				
PXP08UID		09	X				
PXP08BTS		1A					X
PXP08IAO		1B	X				
PXP08IAB		1C		X			
PXP08PRG		1E	X				
PXP08ROO		1F	X				
PXP08DPT		20	X				
PXP08BLD		21	X				
PXP08CON		22	X	X	X	X	X
PXP08ROL		23		X			
PXP08IBT		24	X	X			
PXP08ROS		25			X	X	
PXP08SOS		26	X	X	X		
PXP08BOS		27	X				
PXP08RPH		28	X				
PXP08RPW		2A		X			
PXP08FB1		2B	X				
PXP08IML		2C	X				
PXP08SPA		2E		X			
PXP08SEU		46	X				
PXP08SEP		47	X				
PXP0CINS	0C	01	X	X	X	X	X
PXP0CIXF		02	X	X	X	X	X
PXP0CBTL		03		X			
PXP0CIOE		07	X	X	X	X	X
PXP0CSNF		08	X				
PXP0CCOR		09	X				
PXP10PSP	10	05	X	X	X	X	X
PXP10SIE		06	X	X	X	X	X
PXP10MST		07	X				

Submitting Output Data

This PUT service spools the submitted records as a queue entry in an output (LST, PUN, or XMT) queue. In your program, you issue output-related PUT-service requests as follows:

- A *PUT-OPEN* request to start the spooling of output:
  - To create a new output queue entry. This is the same as for the opening of a job-related PUT service. For details, see “Starting a PUT Service for a Job or a Jobstream” on page 106.
  - To restart an existing queue entry. For details, see “Requesting a Restart” on page 128.
  - To append output to an existing queue entry. For details, see “Appending Output to an Existing Spool File” on page 131.
  - To specify Output Parameter Text Blocks (OPTBs). For details, see “Output Parameter Text Blocks (OPTBs)” on page 132.
  - To specify keyword OPTBs. For details, see “Specifying Keyword OPTBs” on page 133.
- One or more *PUT-SPOOL data* requests to have VSE/POWER spool the submitted output. This is the same as for the submission of job-related spool data; for details, see “Issuing a PUT-SPOOL-Data Request” on page 111.
- A *PUT-CLOSE* request to end the submission of output:
  - If there is no need to add additional spool data later on – this is the same as for the closing of a job-related PUT service; for details, see “Issuing a PUT-CLOSE-Service Request” on page 111.
  - If additional spool data is to be added later on – this is discussed under “Appending Output to an Existing Spool File” on page 131.
- A *PUT-QUIT* request to indicate that no further data is to be submitted and that the output so far spooled is not to be included in a VSE/POWER output queue. This is the same as for a QUIT request during the spooling of job-related data; for details, see “Ending the PUT Service for Jobs” on page 112.
- A *PUT-OUTPUT-SEGMENTATION* request. For details, see “Requesting Output-Segmentation” on page 125.
- A *PUT-CHECKPOINT* request. For details, see “Requesting a Checkpoint for PUT Services for Output” on page 127.
- A *PUT-RESTART* request. For details, see “Requesting a Restart” on page 128.
- A *GET-OPTB* request. This is the same as for a Get-OPTB request during Get service processing; for details see “Issuing a Get-OPTB Request” on page 96.
- A *MODIFY-OPTB* request. This is the same as for a Modify-OPTB request during GET service processing; for details see “Issuing a Modify-OPTB Request” on page 97.

There is one major difference between this service processing and the submission of a job: for the spooling of output, a number of the fields of the required SPL may have to be set up by your program.

To accomplish this, you should:

1. In your program, code the PWRSPL macro with TYPE=GEN or TYPE=UPD and specify the operands

   JOBN=..., to provide the name of the output entry

   USERID=...,which feeds both the FROM user ID (SPLGUS) and the TO user ID (SPLDTUID).

2. Use the available SPL DSECT to access the SPL.

For a list of the applicable SPL fields, see Table 36. For the lengths and data types of these fields, see the SPL DSECT that you get by a PWRSPL macro with TYPE=MAP; this DSECT gives additional explanation. The DSECT is listed under “Spool-Access Support Parameter List (PWRSPL DSECT)” on page 231.

Table 36. SPL Fields Applicable to a PUT-Output Service Request

Name of Field	Applies to LST	Applies to PUN	Purpose/Contents
SPLORCFM [3]	M	M	Record format
SPLGCL	O	O	Job (output) class
SPLGPW	O	O	Password
SPLGOPT2	O	O	Option byte (set SPLGO2BT if trailing blanks of records should not be truncated during spooling) [4]
SPLDDP	O	O	Output local disposition
SPLDPR	O	O	Output priority
SPLDSID	O	O	Output-system ID
SPLDMOHP	O	O	Protect option: Hold (with disposition Y) when print/punch fails
SPLDUI [1]	O	O	User information
SPLDTNN	O	O	Name of destination node
SPLDTUID [5]	O	O	Name of destination user
SPLDPRGN	O	O	Programmer name
SPLDROOM	O	O	Room number
SPLDDEPT	O	O	Department number
SPLDBLDG	O	O	Building number
SPLDCREC	O	O	PUT-open restart record number
SPLDLREC	O	O	Maximum record length
SPLONCPY	O	O	Number of copies
SPLOCOMP	O		Name of compaction table
SPLOFORM	O	O	Form number
SPLOEWTR	O	O	External writer subsystem
SPLOFCB	O		Name of FCB-image phase
SPLOUCB	O		Name of UCB-image phase
SPLOUCBO	O		UCB options
SPLONSEP [2]	O	O	Number of separator pages/cards
SPLOTDP	O	O	Output transmission disposition
SPLEOPOF	O	O	Offset to OPTB area
SPLEOPLN	O	O	Length of passed OPTBs
SPLEOPTB	O	O	First (or only) OPTB
3200/3800 Specifications (bit SPL3F138 must be set if any 3200/3800 option is specified)			

Table 36. SPL Fields Applicable to a PUT-Output Service Request (continued)

Name of Field	Applies to LST	PUN	Purpose/Contents
SPL3TAB1	O		Character-arrangement table 1
SPL3TAB2	O		Character-arrangement table 2
SPL3TAB3	O		Character-arrangement table 3
SPL3TAB4	O		Character-arrangement table 4
SPL3MODF	O		Copy-modification phase
SPL3CCHR	O		Character-arrangement table for copy-modification text
SPL3CPYG	O		Copy-group values
SPL3FLSH	O		Flash-ID
SPL3FLCT	O		Number of copies to be flashed
SPL3FLG1	O		Options byte (bit SPL3F138 must be set if any 3200/3800 option is specified)
SPLXDIST	O	O	Distribution code
SPLXFLG1	O	O	Extended flag byte 1 (set SPLX1SNO, if output NOT to be spool access protected (corresponding to SECAC=NO))
SPLXPMDE	O	O	Processing mode (PRMODE)
SPLXEXPD	O	O	Queue entry expiration days
SPLXEXPH	O	O	Queue entry expiration hours

[1] Any hexadecimal value may be used. The system uses the OR operation which converts characters with a hexadecimal 40 (X'40') value. Because problems may arise when displaying non-printable characters on a console or printer, it is strictly recommended to use only hexadecimal values which, after conversion, represent printable characters.

[2] If SPLONSEP contains X'40', VSE/POWER will use the number of separator cards/pages specified in the JSEP operand of the VSE/POWER generation macro. Valid specifications are hexadecimal numbers 0-9. If nothing is specified, the PWRSPL macro defaults to 0.

[3] This field allows to select one of the record formats SCS, BMS, 3270, CPDS, ESC, ASA, and MCC. When MCC or ASA format is specified, then later during spooling, VSE/POWER will not only accept the specified control character type, but also CPDS type intermixed.

[4] For details on blank truncation, refer to “Recording of Spooled Data on the Data File” in the *VSE/POWER Administration and Operation*, SC34-2625.

[5] For immediate local printing, specify user ID "R000"; for details, refer to *VSE/POWER Administration and Operation*, SC34-2625.

Legend: M = Mandatory; O = Optional

Format of Spool Output Records

Every output record that is to be spooled by VSE/POWER must have an 8-byte record prefix as shown in Table 24 on page 77. The record prefix must be updated for the following fields (refer to“Spool-Access Support Parameter List (PWRSPL DSECT)” on page 231):

RECCCODE
: carriage control character

RECTYPE
: record type

RECLNGTH
: length of the subsequent logical record

The specified control character should correspond to the general record format of the output queue entry as preselected at PUT-OPEN in the mandatory SPL field SPLORCFM (see Table 36 on page 118). VSE/POWER does not verify the validity of the corresponding character when specified in any record prefix. The coding sequence at label FILLBUFFO in Chapter 13, "Spool-Access Support Programming Example," on page 271 shows how to set up the record prefix for output records.

Only for spooling of CPDS (Composed Page Data Stream) does VSE/POWER impose the following rules:

1. When SPLORCFM specifies SPLORAPA, meaning CPDS format, then *all* spooled output records must be identified as CPDS type by their prefix.
2. When SPLORCFM specifies MCC or ASA, the spooled output records may be intermixed with CPDS records.

CPDS records must be identified either by the carriage control character (RECCCODE=X'5A') and/or the record type (RECTYPE=RECTCPDS). If only one of the two identifications is specified, VSE/POWER enforces the missing one correspondingly. For further details on handling of this record type, refer to *VSE/POWER Administration and Operation*, SC34-2625.

Spooling of Records with Carriage Control Character X'FE'

The carriage control characters X'FF', X'FE', and X'FD' are reserved for use by VSE/POWER. When spooling records with one of these carriage control characters using the PUT service of the spool-access support, the spooling of the record is rejected with a return and feedback code (08/2F=PXP08ICC).

Using the option CTLREC during a GET service, the retrieved output may contain records with a carriage control character X'FE'. In order to spool back such an retrieved output 'as is' (which means without removing the records with a carriage control character X'FE'), the option SPLGO2FE within SPLGOPT2 must be used during the PUT service.

Note: A record with carriage control character X'FE' indicates the beginning of a new page. VSE/POWER creates an FE-record and increments the page count at output spooling time, when more records are spooled than fit onto one page (according to the value specified in an FCB or LTAB) and the user program did not explicitly start a new page via a skip-to-channel-one command. Spooled FE-records are also used when restart on page boundary is requested.

Page and Line Counts

For the records being spooled, VSE/POWER maintains page and line counts depending on the record type. The table in Table 37 on page 121 shows how VSE/POWER maintains these counts.

Table 37. Line Counts as Maintained by VSE/POWER

Type of Records	**Line Count**	**Page Count**[1]
With ASA	Incremented for every record.	Updated in accordance with carriage-control characters.
With MCC	Updated in accordance with carriage-control characters. X'00' and X'01' (write-no-space) is counted as a line.	Updated in accordance with carriage-control characters.
BMS, 3270 mapping	Incremented for every record.	Incremented for every page.
CPDS	Incremented for every record.	Incremented for every CPDS page
All others	Incremented for every record.	Set to 1
CPDS intermixed with records having ASA or MCC.	Incremented for a CPDS record. For non-CPDS records, see ASA- or MCC-type records, above.	See CPDS, ASA or MCC type records above.

[1] Is set to 1 if, at the end of spooling, this count is still zero and the line count is 1 or greater.

[2] The page count is derived from the structured field identifiers:

- BPG (Begin Page)
- IDM (Invoke Data Map)
- IMM (Invoke Media Map)

Their sequence and combination with non-CPDS records increments the page count such that it comes as near as possible to the actual number of pages printed for this queue entry by the Print Support Facility (PSF/VSE). For structured field identifiers, refer to *PSF Data Stream Reference*, SH35-0073. Refer also to *VSE/POWER Administration and Operation*, SC34-2625.

VSE/POWER Account Records

VSE/POWER performs accounting for submitted output as follows:

- For output without a restart or a later expansion by an append operation, VSE/POWER's spool-record count is the same as for the output of a job submitted from a unit record input device.
- For output to be appended to an existing queue entry, VSE/POWER builds an extra set of spool-access account records:
  - Every time records are submitted to be appended, and
  - Only for the records submitted during the append operation.
- For output involving a restart, VSE/POWER counts the spooled output records only once. Assume, your program:

1. Submits 1000 records for spooling
2. Requests a restart at record position 901
3. Submits another 200 records before it issues a CLOSE request

VSE/POWER's record count then is 1100 records.

Verification SPLs

VSE/POWER returns a verification SPL in the following cases:

1. After having successfully processed a PUT-OPEN request.

   This SPL contains the same information that your program supplied in the request SPL, plus default values assigned by VSE/POWER, for values not specifically supplied, the job number, and the queue entry number.
2. At the end of data submission for the queue entry.

   VSE/POWERpasses this verification SPL in response to your PUT-CLOSE request when submission of output is complete or after having completed a segmentation request. In addition to the information supplied by your program, the SPL includes:
   - All of the VSE/POWER-generated job information, such as job number, job suffix (segment number), and queue entry number.
   - The default values used by VSE/POWER for values not specifically supplied by your program.
   - Statistics such as the total number of records spooled for your output.

   Checking this SPL can be of significance for spooling output. You need, for example, the VSE/POWER-assigned job number if data is to be appended to this queue entry or if spooling is to be restarted.

Handling an Abnormal-End Condition During PUT-SPOOL

If an abnormal-end occurs while VSE/POWER spools the output data, VSE/POWER's actions are as follows:

- For an abnormal end of your program or of the VSE/POWER service task:

  If the *output is checkpointed*, VSE/POWER retains the queue entry's spool data up to the last recorded checkpoint. The queue entry's disposition is X to avoid that another task can process the entry.

  If the *output is not checkpointed*, VSE/POWER deletes the currently processed queue entry, except as indicated below:
  – The failure occurred after successful completion of a PUT-OPEN-RESTART request by VSE/POWER. In this case, the previously submitted data up to (but not including) the restart record still exists in the affected queue entry.

    VSE/POWER retains this queue entry with a disposition of X, and your program can set up the requested restart once more.
  – The failure occurred after successful completion of a PUT-OPEN-APPEND request by VSE/POWER. In this case, the data previously submitted (prior to the open-append request) still exists in the affected queue entry.

    VSE/POWERretains this queue entry with a disposition of X, and your program can set up a restart request.

  How to perform a restart is discussed under “Requesting a Restart” on page 128.
- For an abnormal end of VSE/POWER or of the z/VSE system:

  During VSE/POWER startup, VSE/POWER searches the queue file for incomplete queue entries.

- If the *queue entry is checkpointed*, VSE/POWER sets the spool pointer immediately behind the record last checkpointed. In addition, it adds the queue entry to the applicable class chain. When VSE/POWER startup is complete, the queue entry is accessible for a restart request from your program or for printing if the central operator alters the entry's disposition.
- If the *queue entry is not checkpointed*, VSE/POWER deletes this queue entry and the related space of the data file.

- For an abnormal end because of an I/O error on the data file:
  - If the *output is checkpointed*, VSE/POWER retains the queue entry's queue record and associated DBLK groups up to the last recorded checkpoint. The queue entry's disposition is set to X to avoid that another task can process the entry.
  - If the *output is not checkpointed*, VSE/POWER deletes the queue entry.

Coding Sequence for PUT-OUTPUT Requests

In general, the coding sequence for output submission is the same as for the submission of a job (or jobs) for queuing in an input queue. Your program issues an OPEN request, followed by a number of PUT-data requests, followed by a CLOSE request and one or more message retrieval requests; your program can issue a QUIT request any time after a PUT-data request is complete. As mentioned earlier, this section deals primarily with output specific PUT requests.

Issuing a CLOSE-Service Request

Refer to Table 38, a coding-sequence diagram for a PUT-output CLOSE request. You issue the request by setting byte PXUACT1 of your program's XPCCB to either of the following:

- The value equated to PXUATEOD – if no additional data is to be appended.
- The value equated to PXUATROE – if additional data is to be appended at a later point in time. Appending additional data is discussed under “Appending Output to an Existing Spool File” on page 131.

Your program may pass the CLOSE request in one of the following ways:

- Together with a null buffer

  If you do this (after having successfully passed a send buffer containing data), your program must:

  1. Set byte PXUBTYP of the XPCCB to zero.
  2. Set up a null buffer by setting IJBXBLN to zero.

     How to set up a null buffer is shown in Chapter 13, “Spool-Access Support Programming Example,” on page 271 at the label GQUIT.
  3. Issue an XPCC FUNC=SENDR request after having set up the request.

- Together with a send buffer containing data records

  These records are the last output records spooled by VSE/POWER for the currently processed queue entry. The coding sequence at the label SDEOD in Chapter 13, “Spool-Access Support Programming Example,” on page 271 shows how to do this.

Table 38. PUT-Output CLOSE Request Sequence

Coding in your application program	Comments
... ...	

Table 38. PUT-Output CLOSE Request Sequence (continued)

Coding in your application program	Comments
WAIT IJBXSECB	Wait for the SENDR ECB to be posted after the PUT-data request that is to precede your CLOSE request.
Check the reason code (in the XPCCB byte IJBXREAS).	
Check the VSE/POWER return and feedback codes (in the XPCCB bytes PXPRETCD and PXPFBKCD, respectively).	
CLOSE request XPCC FUNC=SENDR,...	You can issue the request either: - With the send buffer containing data or an SPL. - With a null buffer being passed.
Check the return codes in register 15 and in the XPCCB (byte IJBXRETC).	
WAIT IJBXSECB	Wait for the SENDR ECB to be posted. VSE/POWER passes a verification SPL to your program's reply buffer.
Check the reason code (in the XPCCB byte IJBXREAS).	
Check the VSE/POWER return and feedback codes (this is the same as above).	
Pick up and evaluate the verification SPL, if necessary.	
End of Service	

- Together with a send buffer containing an update SPL

  If you do this (after having successfully passed a send buffer containing data), your program must:

  1. Set byte PXUBTYP of the XPCCB to PXUBTSPL.
  2. Build the SPL in (or move it to) your program's send buffer.

     VSE/POWER analyzes the SPL and updates the control values for the currently processed output queue entry. This SPL is some kind of a last-minute change of the queue entry's job characteristics. However, VSE/POWER verifies only those of this SPL's fields which are listed in Table 39 on page 125; it ignores all other specifications passed by your program.
  3. Issue an XPCC FUNC=SENDR request after having set up the request.

VSE/POWER returns a verification SPL to your program's reply buffer. This SPL includes the VSE/POWER assigned job number and queue entry number.

If the output's destination is another node, VSE/POWER spools this output into the XMT queue rather than into the local LST or PUN queue.

Table 39. Update SPL Fields Verified by VSE/POWER

Field Name	Purpose/Contents
SPLGJB	The job name
SPLGCL	The desired output class
SPLDDP [1]	The output local disposition
SPLDPR [1]	The desired output priority
SPLDSID	The ID of the system that is to process the output (applies to a shared spooling environment; only the ID of the z/VSE system is valid).
SPLDTNN	The name of the destination node
SPLDTUID	The destination (remote) user ID
SPLONCPY	The number of desired copies
SPLOFORM	The form number to be used
SPLOTDP	The output transmission disposition
[1] Can be updated only if the submitted output is to be spooled into a local queue.	

Requesting Output-Segmentation

Refer to Table 40, a coding-sequence diagram for an output-segmentation request.

Table 40. Segmentation During PUT-Output Processing Sequence

Coding in your application program	Comments
... ...	
WAIT IJBXSECB	Wait for the SENDR ECB to be posted after the PUT-data request that is to precede your output-segmentation request.
Check the VSE/POWER return and feedback codes (in the XPCCB bytes PXPRETCD and PXPFBKCD, respectively).	
Segmentation request XPCC FUNC=SENDR,...	You can issue the request either: • With a null buffer being passed • With the send buffer containing data • With the send buffer containing an SPL
Check the return codes in register 15 and in the XPCCB (byte IJBXRETC).	
WAIT IJBXSECB	Wait for the SENDR ECB to be posted. VSE/POWER returns a verification SPL to your program's reply buffer.
Check the reason code (in the XPCCB byte IJBXREAS).	
Check the VSE/POWER return and feedback codes (this is the same as above).	
Pick up and evaluate the verification SPL, if necessary.	

Table 40. Segmentation During PUT-Output Processing Sequence (continued)

Coding in your application program	Comments
PUT-Data Request XPCC FUNC=SENDR,...	Continue after having filled your program's send buffer again. The first data record in your program's send buffer goes into the new output segment.
... ...	

Your program can request output-segmentation at any time after successful completion of a PUT-data request. You code this request in your program by setting byte PXUACT1 of the XPCCB to the value equated to PXUATSGM. You pass this request to VSE/POWER in one of the following ways:

- Together with a null buffer

  If you do this, your program must:

  1. Set byte PXUBTYP of the XPCCB to zero.
  2. Set up a null buffer by setting IJBXBLN to zero.

     How to do this is shown in Chapter 13, “Spool-Access Support Programming Example,” on page 271 at the label GQUIT.
  3. Issue an XPCC FUNC=SENDR request after having set up the request.
- Together with a send buffer containing data records

  This causes VSE/POWER to include the buffer's contents in the currently processed output segment. The contents of the next buffer that your program passes to VSE/POWER becomes part of the newly created output segment.
- Together with a send buffer containing an update SPL

  If you do this, your program must:

  1. Set byte PXUBTYP of the XPCCB to PXUBTSPL.
  2. Build the SPL in (or move it to) your program's send buffer.

     VSE/POWER analyzes the SPL and updates the control values for the currently processed output queue entry. This SPL is some kind of a last-minute change of the queue entry's job characteristics. However:

     – VSE/POWER verifies only those of the SPL's fields which are listed in Table 39 on page 125; it ignores all other specifications passed by your program.

     – Any changed (or new) specifications that your program supplies in this update SPL are used by VSE/POWER also for the subsequent segment(s). If this is not desirable, your program has to pass another update SPL at the end of the next segment.
  3. Issue an XPCC FUNC=SENDR request after having set up the request.

Just like for a CLOSE request, VSE/POWER returns a verification SPL after having successfully queued the segment. This SPL gives the VSE/POWER assigned job-suffix (segment) number. VSE/POWER is then ready to accept further output for spooling into a new output segment.

The coding sequence in Chapter 13, “Spool-Access Support Programming Example,” on page 271 includes an output-segmentation request at the label PUTB2.

Requesting a Checkpoint for PUT Services for Output

Consider requesting checkpoints to “save” the processing of records already passed to VSE/POWER should an abnormal-end condition occur. Your program can issue a checkpoint request before the first PUT-data request and after successful completion of any subsequent PUT-data request.

In processing a checkpoint request, VSE/POWER marks the queue entry as having been checkpointed and returns a checkpoint-response record. This response record contains the VSE/POWER-recorded number of the record spooled for the queue entry just before the checkpoint was taken. For the layout of a checkpoint-response record, see the DSECT at the label PXCRDSECT.

The number of the checkpointed record may not be the same as the number of this record according to your program's own record count. Therefore, your program should:

1. Relate the checkpoint record number to the corresponding record number of the program's own count.
2. Save this relation for a later restart, should this become necessary.

By relating this number to your own program's record count, you can synchronize your program's output with the record count maintained by VSE/POWER.

Refer to Table 41, a coding-sequence diagram for a checkpoint request. In your program, you code this request as follows:

1. Set byte PXUACT1 of the XPCCB to the value equated to PXUATCHK.
2. Make a reply buffer available.
3. Pass the request to VSE/POWER. To do this, issue an XPCC FUNC=SENDR request with either of the following:
   - Data contained in your program's send buffer. In this case, VSE/POWER spools that buffer's contents first and then processes the checkpoint request.
   - A null buffer. This requires that your program:
     a. Sets byte PXUBTYP of the XPCCB to zero.
     b. Sets up a null buffer (field IJBXBLN set to zero). For information on how to set up a null buffer, see Chapter 13, “Spool-Access Support Programming Example,” on page 271 – at label GQUIT, for example).

Table 41. Checkpoint for PUT-Output Processing Sequence

Coding in your application program	Comments
... ...	
WAIT IJBXSECB	Wait for the SENDR ECB to be posted after the PUT-data request that is to precede your checkpoint request
Check the VSE/POWER return and feedback codes (in the XPCCB bytes PXPRETCD and PXPFBKCD, respectively).	

Table 41. Checkpoint for PUT-Output Processing Sequence (continued)

Coding in your application program	Comments
Checkpoint request XPCC FUNC=SENDR,...	You can issue the request either: • With the send buffer containing data • With a null buffer being passed.
Check the return codes in register 15 and in the XPCCB (byte IJBXRETC).	
WAIT IJBXSECB	Wait for the SENDR ECB to be posted. VSE/POWER passes a checkpoint response record to your program's reply buffer.
Check the reason code (in the XPCCB byte IJBXREAS).	
Check the VSE/POWER return and feedback codes (this is the same as above).	
PUT-data request XPCC FUNC=SENDR,...	Continue after having filled your program's send buffer again.
End of Service	

Requesting a Restart

VSE/POWER permits your program to request a PUT-RESTART as follows:

- *During PUT-SPOOL* processing for output, behind a previously spooled record.

  This restart causes the specified restart record and all subsequent records spooled previously to be overwritten.

  A restart during processing can be risky. Your program's record count (if maintained) may be different from that of VSE/POWER because VSE/POWER inserts an additional record whenever a write-and-skip to channel 1 occurs. “Requesting a Checkpoint for PUT Services for Output” on page 127 indicates how your program can use VSE/POWER's checkpoint-response records to keep track of suitable restart points. See “Restarting During PUT-Output Processing” on page 129.
- Together *with* a PUT-OPEN request for output for an existing queue entry.

  As the restart point, you can specify 0 (or nothing). In this case, VSE/POWER sets its restart pointer immediately behind the last record in the queue entry's last used DBLK of the data file. For a checkpointed queue entry with disposition X, this is the record last checkpointed by VSE/POWER.

  Specifying 0 may be risky. If a system or program failure occurs after VSE/POWER has passed a recorded checkpoint and before your program could record this checkpoint, then VSE/POWER and your program are not synchronized.

To avoid problems, you can specify a suitable restart point as recorded by your program. VSE/POWER indicates in its verification SPL the corresponding, checkpointed record count.

In case of a restart, VSE/POWER examines a specified restart point. If this point:

– Is higher than the last recorded checkpoint, VSE/POWER accepts this restart point as specified.
– Is equal to or lower than the last recorded checkpoint, VSE/POWER lowers the checkpoint value to the restart value, minus 1, and notifies your program of the change (return/feedback code = PXPRCOK/PXP00CIA).

See “Restarting with an OPEN Request” on page 130.

Note: Because PUT-RESTART involves a GET operation for an existing spool entry, restart may be denied if Spool Access Protection is active. See “Scope of GET/CTL Access to Queue Entries” on page 61.

Restarting During PUT-Output Processing

If, in its restart control record, your program specifies a restart record number lower than or equal to the logical record last checkpointed, then VSE/POWER:

1. Positions the spool pointer as requested, just as if the queue entry were not checkpointed.
2. Records the specified restart record number (minus one) as the new checkpoint-record number.

   VSE/POWER returns to your program a checkpoint-response record together with applicable return and feedback codes. The response record confirms to your program the newly recorded checkpoint. For the layout of a checkpoint-response record, see the DSECT generated by PWRSPL TYPE=MAP at the label PXCRDSCT.

Refer to Table 42, a coding-sequence diagram for a restart request.

Table 42. Restart for PUT-Output Processing Sequence

Coding in your application program	Comments
... ...	
WAIT IJBXSECB	Wait for the SENDR ECB to be posted after the PUT-data request that precedes your restart request.
Check the VSE/POWER return and feedback codes (in the XPCCB bytes PXPRETCD and PXPFBKCD, respectively).	
Restart request XPCC FUNC=SENDR,...	Your program's send buffer must contain a restart control record.
Check the return codes in register 15 and in the XPCCB (byte IJBXRETC).	
WAIT IJBXSECB	Wait for the SENDR ECB to be posted. VSE/POWER may pass a checkpoint-response record to your program's reply buffer.

Table 42. Restart for PUT-Output Processing Sequence (continued)

Coding in your application program	Comments
Check the VSE/POWER return and feedback codes (this is the same as above).	
Pick up and evaluate the check-point response record, if this is applicable.	
PUT-data request XPCC FUNC=SENDR,...	At this point, the coding sequence is the same as for the submission of data records for spooling.
End of Service	

To set up and pass the request to VSE/POWER, your program must:

1. Set byte PXUBTYP of the XPCCB to the value equated to PXUBTCTL. This indicates that your program's send buffer contains a control record.
2. Set up a restart control record in your program's send buffer. For the layout of this record, see the Dsect at label PXRSDSCT. The record specifies the number of the logical record at which output spooling is to be resumed.
3. Issue an XPCC macro with FUNC=SENDR.

Restarting with an OPEN Request

This kind of a restart applies if output spooling is to be restarted because, for example, an abnormal-end condition had occurred.

A PUT-OPEN-RESTART request is possible if the following is true:

- The applicable queue entry is queued with one of the dispositions D, H, K, L, and X.
- The requestor is the owner (originator) of the queue entry.

If your program does not pass a restart-record number, then:

- For a queue entry with disposition X, VSE/POWER positions the spool pointer behind the entry's last checkpointed record.
- For a queue entry with a disposition other than X, VSE/POWER positions this pointer to the end of the entry's data file.

If your program passes a restart-record number, it should provide for a routine verifying that VSE/POWER's record count and your program's record count are synchronized. How you can do this is indicated under “Restarting During PUT-Output Processing” on page 129.

Your program requests the desired restart by issuing an XPCC macro with FUNC=SENDR and passing to VSE/POWER a restart SPL (MODE=RESTART specified in the PWRSPL macro). In the SPL, certain fields are to be updated as listed in Table 43 on page 131. VSE/POWER confirms the request in the same way as it confirms a normal open PUT-service request: by passing a verification SPL to your program.

Table 43. SPL Fields to be Updated – Open-Restart Request for Output

Name of Field	Applies to LST	PUN	Purpose/Contents
SPLGFB1	M	M	Set restart function (SPLGF1RS)
SPLGCL	M	M	Job (output) class
SPLGJB	M	M	Job name
SPLGJN	M	M	Job number
SPLGUS	M	M	User ID
SPLGQI	M	M	Queue ID
SPLGRQB	M	M	Request type (PUT)
SPLGJS	O	O	Job suffix
SPLGOPT	O	O	Set no-wait option
SPLGPW	O	O	Password
SPLDCREC	O	O	PUT-open restart record number
Legend: M = Mandatory; O = Optional			

Appending Output to an Existing Spool File

VSE/POWER permits additional data to be appended to (added at the end of) an existing output queue entry if your program:

1. Is the owner (originator) of this queue entry.
2. Closed the original spool request with the append-option bit PXUATROE set in byte PXUACT1 of its XPCCB.
3. Issues a PUT-OPEN APPEND request (which re-initiates PUT-service processing for the queue entry) by passing to VSE/POWER an SPL that specifies the append option. This SPL should contain the values passed by VSE/POWER in its original verification SPL in the fields listed in Table 44. You may find it convenient to have your program save the verification SPL and use it as request SPL for the append request.

Table 44. SPL Fields to be Updated – Open-Append Request for Output

Name of Field	Applies to LST	PUN	Purpose/Contents
SPLGFB1	M	M	Set append function (SPLGF1AP)
SPLGCL	M	M	Job (output) class
SPLGJB	M	M	Job name
SPLGJN	M	M	Job number
SPLGUS	M	M	User ID
SPLGQI	M	M	Queue ID
SPLGRQB	M	M	Request type (PUT)
SPLGJS	O	O	Job suffix (segmented number)
SPLGOPT	O	O	Set no-wait option
SPLGPW	O	O	Password
Legend: M = Mandatory; O = Optional			

VSE/POWER confirms the request in the same way as it confirms a normal open PUT-service request: by passing a verification SPL to your program. After having passed this SPL, VSE/POWER is ready to accept PUT-data requests from your program.

Note: Because appending involves a GET operation for an existing spool entry, the append may be denied if Spool Access Protection is active. See "Scope of GET/CTL Access to Queue Entries" on page 61.

Output Parameter Text Blocks (OPTBs)

VSE/POWER allows you to specify one or more Output Parameter Text Blocks (OPTBs) when you describe the characteristics of the output queue entry passed to VSE/POWER. VSE/POWER allows you to specify standard and keyword OPTB's. For more information on OPTB's, refer to the description of the DEFINE autostart statement in *VSE/POWER Administration and Operation*, SC34-2625.

Specifying Standard OPTBs

An OPTB (also named Output Procesing Text Unit, or OPTU) represents the keyword in an * $$ LST or * $$ PUN statement which you define in an autostart DEFINE statement.

An OPTB is structured as a sequence of text units. The number and sequence of text blocks is arbitrary. The format of the OPTBs is shown in Figure 3.

	ID	CC	LL	Data element	LL	Data element	//
Bytes:	2	2	2	n	2	n	

ID Registered (unique) keyword ID

CC Count of the data elements supplied for the keyword parameter. The valid range is from 0 to 16,383. A count of 0 indicates either a missing positional or defaulted parameter. In this case, no data elements should follow the count field.

LL Length of the data element (keyword parameter value). The valid range is from 0 to 16,383. A length of 0 indicates a null value.

Figure 3. Standard OPTB Format

This format is called **standard** to distinguish it from keyword OPTBs.

For example:

001F	0001	0004	HUGO

causes VSE/POWER to check this standard OPTB against the DEFINE autostart statement for PAGEDEF.

Note: OPTBs (also called OPTUs) are spooled by VSE/POWER and other NJE components in the 'Output Processing Section' of the NJE Data Set Header Record (DSHR). How to locate OPTUs in such a section can be seen in the "Sample of a PNET Receiver Exit Routine" of the *VSE/POWER Networking*, SC34-2603.

Specifying Keyword OPTBs

In your program you can also specify **keyword OPTBs** which pass a user keyword and its values directly to VSE/POWER, without knowledge of the OPTB structure, according to Network Job Entry (NJE) definitions. VSE/POWER matches the received keyword against the specifications of the corresponding DEFINE autostart statement. VSE/POWER creates an OPTB according to the user's request and includes it in the DSHR record.

The format of the keyword OPTBs is shown below, in Figure 4.

	ID	CC	LL	keyword={value|(value,...)}
Bytes:	2	2	2	n

ID ID must be X'0000'.

CC CC must be X'0001'.

LL Length of the keyword and value. A length of 0 indicates that no keyword has been specified. the maximum length is 16.383.

Figure 4. Keyword OPTB Format

For example:

0000	0001	000C	PAGEDEF=HUGO

causes VSE/POWER to build an OPTB according to the DEFINE autostart statement for PAGEDEF.

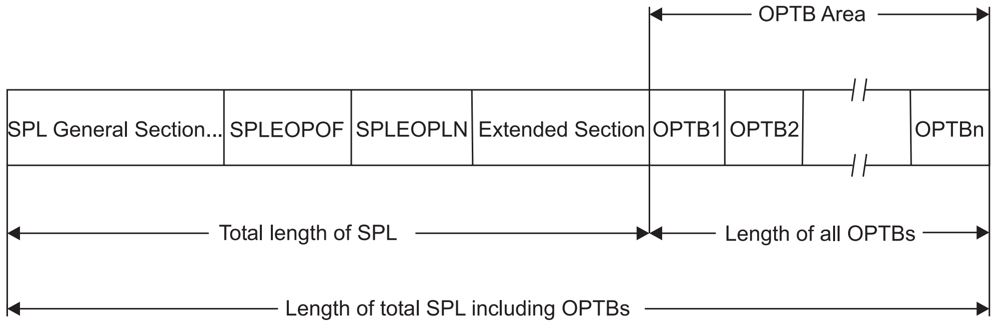
Passing OPTBs to VSE/POWER

You can pass both the standard and the keyword OPTB to VSE/POWER as an appendage of the SPL at PUT OPEN time. You may pass them in the same SPL, but all keyword OPTBs have to precede the standard OPTBs; otherwise, they will be flagged as 'invalid standard' OPTBs.

There are the following restrictions:

- When two or more keyword OPTBs specify the same user keyword, only the last specification becomes effective. The same is true for equal parameters of a * $$ LST or * $$ PUN statement.
- When two or more standard OPTBs are passed with the same OPTB-Id, they are rejected by the return code PXP08DOP.
- When a keyword is specified both by a keyword OPTB and by a subsequent standard OPTB, then duplicate OPTBs are created in the DSHR record.

As shown in Figure 5 on page 134, the SPL contains two 2-byte fields, indicating the total length of the OPTB area and the offset to this area. A length of zero (in field SPLEOPLN) indicates that no such area exists.



Where: SPLEOPOF = Two-byte field containing the offset from the beginning of the SPL to the OPTB area
SPLEOPLN = Two-byte field containing the length of the OPTB area

Figure 5. SPL Format

If one or more standard OPTBs are appended to the SPL, VSE/POWER checks the OPTBs for correct specification. A standard OPTB representing a keyword which is not defined within VSE/POWER is taken as is (refer to *VSE/POWER Administration and Operation*, SC34-2625). If all standard OPTBs are valid, VSE/POWER builds an output processing section and includes it in the data set header record.

For a keyword OPTB, VSE/POWER checks the keyword value against the definition made with a DEFINE statement. If the value is correct, VSE/POWER builds an OPTB and includes it in the data set header record. If no DEFINE statement for the keyword is available, VSE/POWER replies with the return code PXP08NDK.

The total length of all OPTBs, including the length of all other sections (such as the general or the 3800 section) present in the DSHR may not exceed 32,760 bytes.

Note: All standard OPTBs which are of the type binary must have the same length as specified in the appropriate DEFINE statement.

Checking the Return Information for a PUT Service Request for Output

For the return information to be checked by your program after an XPCC request, refer to "XPCC" on page 212.

For every PUT-output service request, your program should also check the return information supplied by VSE/POWER. Provide for this checking after your program's SENDR ECB has been posted.

Table 45 on page 135 lists the return and feedback codes that VSE/POWER may supply when it processes a PUT-service related request for the submission of output data. The list is ordered in ascending order by code values; it relates the codes to the applicable request types; it gives the names that are equated to the feedback codes.

A complete list of the VSE/POWER return and feedback codes is given in the DSECT PXPUSER, which the assembler generates for a PWRSPL TYPE=MAP macro. You find the return codes at label PXPRETCD and the feedback codes at label PXPFBKCD.

For more information on the subject see Chapter 14, "Return and Feedback Codes and Their Meanings," on page 297.

Table 45. Return and Feedback Codes for PUT-Output Service Requests

			Request Type									
Mnemonic	**Ret. Code**	**Fdbk Code**	**PUT Open**	**PUT Data**	**Check point**	**Re-start**	**Seg-ment**	**CLOSE**	**QUIT**	**Get Msg**	**Get OPTB**	**Mod. OPTB**
PXP00OK	00	00	X	X	X	X	X	X	X	X	X	X
PXP00EOD		01								X		
PXP00NRS		03			X		X	X	X			
*PXP00RTR		04		X				X				
PXP00ZBF		05		X								
PXP00CIA		06				X						
PXP04NOF	04	01	X									
PXP04JOP		02	X									
PXP04IDP		05	X									
PXP04RER		06				X						
PXP04SOD		08	X	X			X	X				
PXP04SOA		09	X	X			X	X				
PXP04BER		0A										X
PXP04NMU		0D	X									
PXP04WDP		0E	X									
PXP04JSR		0F	X									
PXP04ONF		11									X	X
PXP04SAC		17	X									

Table 45. Return and Feedback Codes for PUT-Output Service Requests (continued)

			Request Type									
Mnemonic	**Ret. Code**	**Fdbk Code**	**PUT Open**	**PUT Data**	**Check point**	**Re- start**	**Seg- ment**	**CLOSE**	**QUIT**	**Get Msg**	**Get OPTB**	**Mod. OPTB**
PXP08SPL	08	01	X									
PXP08REQ		02	X									
PXP08JNM		05	X				X	X				
PXP08QID		06	X									
PXP08CLS		07	X				X	X				
PXP08PWD		08	X									
PXP08UID		09	X									
PXP08RFM		0A	X									
PXP08DSP		0B	X				X	X				
PXP08PRY		0C	X				X	X				
PXP08SID		0D	X				X	X				
PXP08TNN		0E	X				X	X				
PXP08TUN		0F	X				X	X				
PXP08FNO		10	X				X	X				
PXP08FCB		11	X									
PXP08UCB		12	X									
PXP08FLH		14	X									
PXP08CPT		15	X									
PXP08CGP		16	X									
PXP08CHR		17	X									
PXP08MOD		18	X									
PXP08CCR		19	X									
PXP08BTS		1A								X	X	
PXP08IAB		1C			X		X	X	X	X		
PXP08ICR		1D			X	X					X	X
PXP08PRG		1E	X									
PXP08ROO		1F	X									
PXP08DPT		20	X									
PXP08BLD		21	X									
PXP08CON		22	X	X	X	X	X	X	X	X		
*PXP08ROL		23		X								
PXP08IBT		24	X	X	X	X	X	X	X	X		
PXP08ROS		25						X	X			
PXP08SOS		26	X	X	X	X	X	X		X		
PXP08BOS		27	X					X	X	X		

Table 45. Return and Feedback Codes for PUT-Output Service Requests (continued)

			Request Type									
Mnemonic	**Ret. Code**	**Fdbk Code**	**PUT Open**	**PUT Data**	**Check point**	**Re-start**	**Seg-ment**	**CLOSE**	**QUIT**	**Get Msg**	**Get OPTB**	**Mod. OPTB**
*PXP08RPW	08	2A		X				X				
PXP08FB1		2B	X									
PXP08IML		2C	X									
PXP08IEX		2D	X									
*PXP08SPA		2E		X				X				
*PXP08ICC		2F		X				X				
PXP08IRR		38									X	X
PXP08IOP		39	X									X
PXP08OLM		3A										X
PXP08DOP		3B	X									
PXP08OTL		3C	X									
PXP08IDH		3D									X	X
PXP08DIS		3E	X									
*PXP08INK		3F	X									
*PXP08NDK		40	X									
*PXP08IDV		41	X									
PXP08IPM		48	X									
PXP0CINS	0C	01	X	X	X	X	X	X	X	X		
PXP0CIXF		02	X	X	X	X	X	X	X	X		
PXP0CBTL		03	X									
PXP0CIOE		07	X	X	X	X	X	X	X	X		
PXP0CSNF		08	X									
PXP0CCOR		09	X									
PXP10PSP	10	05	X	X	X	X	X	X	X	X		
PXP10SIE		06	X	X	X	X	X	X	X	X		
PXP10MST		07	X									

* Along with these return and feedback codes, VSE/POWER returns in the user data field PXPROFF an offset value within the user's send buffer. With this offset, the failing record or wrong keyword OPTB can be found in the send buffer of the user's program. Add the offset to the beginning of the send buffer. For PXP00RTR the offset of the last truncated record is returned.

Chapter 10. GCM - Retrieving Job Event and Output Generation Messages, Inquiring eXtended Event Messages

The GCM (Get Completion Messages) service returns job event and output generation messages to your program from jobs which have been submitted to and processed by VSE/POWER. Any user-written application program can, therefore, retrieve these job event and output generation messages but only if the jobs are submitted via the spool-access support with the "queue-event-message" option set in the SPL (spool parameter list). For example, when this SPL option is set, VSE/POWER will collect the message in fixed format rather than issuing the message to, for example, the system console.

VSE/POWER can queue job completion messages (JCMs), job generation messages (JGMs), and output generation messages (OGMs), which are generated in the following cases:

- Job completion message 1Q5DI is produced when a job has been completed.
- Job generation message 1Q5HI is produced when a job has generated another job in the form of punch output with DISP=I.
- Output generation message 1Q5RI is produced each time when a job has created LST or PUN output and this output is ready for processing.

Application program can use fixed format messages in such examples as for determining whether a job was cancelled or ended abnormally, to check job output stream, as well as for other purposes.

The NTFY=YES|(nodeid,userid) operand in the * $$ JOB statement can be used in conjunction with the SPL option when the job is submitted. This means, if you select both options for a job at the same time, two messages will be generated. The 1Q5DI message due to the NTFY operand is routed to the destination specified in the operand. The SPL option, on the other hand, produces a fixed format job event message which is recorded for your application program for later retrieval.

There is no similar support for the job generation message 1Q5HI and output generation message 1Q5RI, meaning within the * $$ JOB statement there doesn't an operands similar to NTFY for these messages. VSE/POWER will collect these fixed format messages for user-written application programs but, for example, doesn't issue them to the system console.

The GCM service also provides XEM (eXtended Event Message) support to application programs. VSE/POWER can generate and queue extended event messages 1Q5XI for an application in the following cases:

- A new queue entry has been created within RDR, LST, PUN, or XMT queue and is ready for processing or has been spooled to a tape.
- An existing queue entry has been altered within a VSE/POWER queue.
- An existing queue entry has been deleted from RDR, LST, PUN, or XMT queue (removed into DEL queue).

Using extended event messages, an application program can check VSE/POWER queues for new and altered entries and can obtain information about deleted entries.

As opposed to other fixed format messages, whose generating period is restricted by a job life time, VSE/POWER produces XEMs independently of any specific job. For both starting XEM generation and extracting messages, extended GCM service (GCM-XEM) is used.

Detailed description of the extended GCM-XEM service can be found in the section "GCM-XEM Service" on page 155 in this chapter.

Destination of Job Event and Output Generation Messages

When you submit a job to VSE/POWER via the spool-access support, some of the specifications you have to code are:

- The application ID by means of the APPL operand in the XPCCB macro
- The USERID operand in the PWRSPL macro.

VSE/POWER uses this information in order to address the message queue in which the fixed format job event and output generation messages are to be queued. Therefore, for each single pair of application ID and PWRSPL user ID a specific message queue can exist while VSE/POWER is up and running.

In order to retrieve the messages resulting from the jobs you submitted, your application program which issues the GCM service has to provide the same user ID and application ID again.

There is another, a higher, level of event messages destination: a system in a Shared Complex and PNET node in a network environment (for details, refer to "Shared Processing" on page 153 and "Networking" on page 153).

The Size of the Message Queue

The system administrator can define the size of a single message queue in the JCMQ=nnn operand in the VSE/POWER SET autostart statement. The message queue can list from 0 to 255 messages. The default setting is set to a maximum of 50 messages queued. However if a message queue size of 0 is specified, then fixed format job event and output generation messages are lost. VSE/POWER does not reserve any storage for event messages unless such messages are generated.

For explanation about the JCMQ=nnn operand, see the description of the SET autostart statement in *VSE/POWER Administration and Operation*, SC34-2625.

If the capacity of your message queue is exhausted, VSE/POWER discards the oldest message in the queue to make room for the next message to be queued. If the message queue size is too small for your needs, VSE/POWER informs your application program and the operator as follows:

- A return and feedback code is passed along with the verification SPL, which VSE/POWER passes to your application after the PUT-OPEN job request (refer to "Requesting Job Event and Output Generation Messages" on page 109).
- Message 1Q4AI appears at the system console at job processing time. This message appears each time a message is discarded or lost in the message queue and when at least 60 seconds are passed since the last appearance of message 1Q4AI.
- The PDISPLAY STATUS command can be used to obtain the maximum number of job event and output generation messages discarded from any message queue.

The message queue size is shown in the statistics status report of the PDISPLAY STATUS command and after issuing a PEND command.

Messages which are not retrieved but remain queued are lost at VSE/POWER shutdown.

Requirements for Requesting the GCM Service

The GCM service uses the same XPCC protocol as the PUT, GET, and CTL services. Refer to the following sections for detail on these services:

- Chapter 6, “Introduction to Spool-Access Support,” on page 57.
- “Setting Up a Communication Path” on page 59.

How to Submit a Job with the 'Queue Event Message' Option

For details, refer to “Requesting Job Event and Output Generation Messages” on page 109.

How to Enable Completion Message Queuing by Command

For details, refer to “Enabling Job Completion Messages by the Release Command” on page 70.

How to Retrieve Job Event and Output Generation Messages

There are two different ways to retrieve job event and output generation messages:

1. The application issues a GCM-OPEN request and VSE/POWER posts the request immediately with or without messages. If there are no messages, then the application can repeat the GCM-OPEN request as many times as needed until VSE/POWER returns messages to the application's receive buffer. To decrease CPU utilization, the application can call the SETIME macro with a subsequent WAIT call or use a previously specified interrupt routine before issuing the GCM-OPEN request again.
2. The application specifies a wait interval in the GCM-OPEN request. VSE/POWER will then wait until the issuance of any message or until the time period is expired. Whereupon, VSE/POWER returns message(s) (if any) to the application's receive buffer, posts the application and cancels the wait interval. Afterwards, the application can set the wait interval again with a new GCM-OPEN request.

The following explanations are common to both retrieval methods. How to specify the wait interval is described under “Optional Specifications Related to the GCM-OPEN Request” on page 148.

Your application program must adhere these steps to retrieve fixed format job event and output generation messages:

1. Issue the relevant XPCC function calls with the same XPCC applid used at PUT-JOB time to identify your program to the system and to establish the connection to VSE/POWER as described under “Setting Up a Communication Path” on page 59.
2. Issue a GCM-OPEN request.
   - Set byte PXUBTYP of the XPCCB to the value of PXUBTSPL. This indicates to VSE/POWER that the send buffer contains an SPL.

- Set up a reply buffer to which VSE/POWER passes the retrieved messages. The size of the reply buffer determines the maximum number of messages which VSE/POWER can pass to your program with its reply. Specify the address of the reply buffer in the XPCCB; for details see Figure 2 on page 58. For details on the length of one message, see Figure 7 on page 143.
- Set up an SPL using the PWRSPL macro with TYPE=GEN to generate or update with TYPE=UPD with the following features:
  - REQ=GCM to signal that a GCM request is issued by your program.
  - The USERID operand you specified at job submission time.
  - Specify a message selection criteria (job name, job number, and message type), as described in "Message Selection Criteria" on page 144.
  - Specify the type of GCM-OPEN service as described under "GCM-OPEN Request Types" on page 145 in SPLGFB1 to inform VSE/POWER what to do with the queued message.
  - Set up optional specifications within the SPL, as described under "Optional Specifications Related to the GCM-OPEN Request" on page 148.
- Establish the SPL as SEND buffer. You must specify the SPL address and length in the XPCCB fields IJBXADR and IJBXBLN.
- Send the SPL to VSE/POWER by means of the XPCC FUNC=SENDR request.

3. After your program has been posted in field IJBXSECB, evaluate the return and feedback codes which VSE/POWER passes to your program in bytes PXPRETCD and PXPFBKCD to decide, for example, if messages are available in your reply buffer (if all messages are retrieved or if there are more messages still waiting for retrieval), or to terminate the GCM service.
4. Process and evaluate the messages contained in your reply buffer, if applicable. To access the message data, see the general message layout in Figure 7 on page 143 and detailed message fields in "Spool-Access Support Parameter List (PWRSPL DSECT)" on page 231.
5. If applicable, your program may issue a GCM subrequest, that is a GCM-MORE or a GCM-REMOVE subrequest, whatever is required. For details see "GCM Subrequests" on page 150.

The following diagram summarizes how the fixed format job event and output generation messages can be retrieved by your application program.

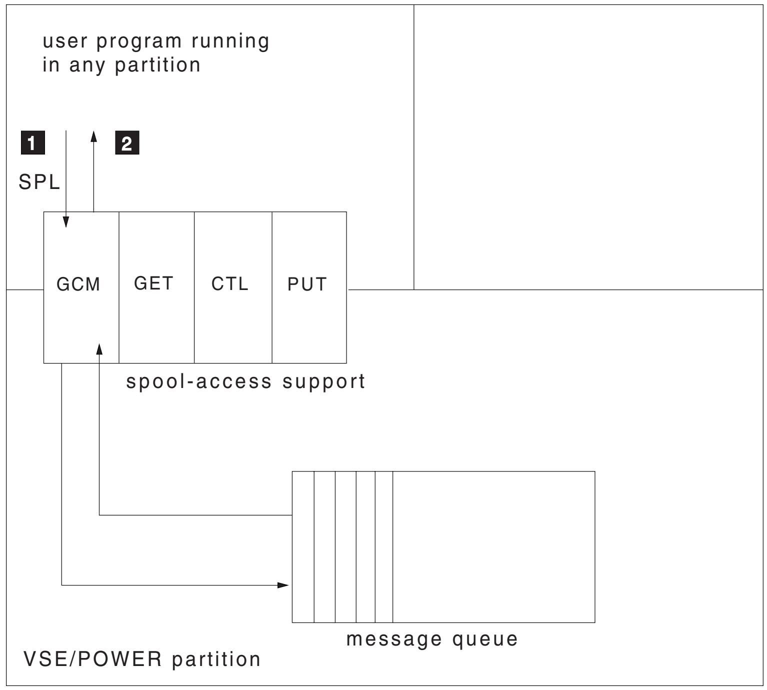


Figure 6. Retrieval of Queued Fixed Format Job Event and Output Generation Messages

1. Your program issues a GCM-OPEN request to retrieve and process fixed format messages under its XPCC applid and SPLGUS userid.
2. VSE/POWER puts the messages into its buffer (counterpart of the program's reply buffer) and informs XPCC interface about its location. XPCC , in its turn, moves messages from this buffer to your program's reply buffer whose address is contained in the XPCCB. From this reply buffer your program can retrieve the messages.

Layout of a Fixed Format Job Event and Output Generation Message

The layout of one message data record in your reply buffer is as follows:

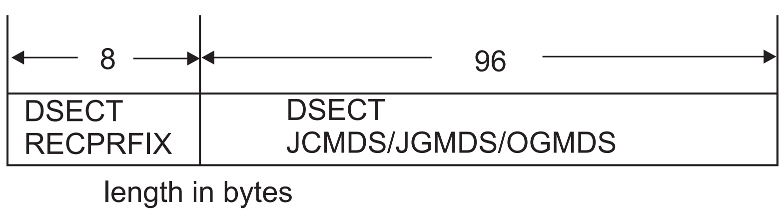


Figure 7. Layout in Bytes of a Fixed Format Job Event and Output Generation Message

The PWRSPL macro provides four DSECTs, namely RECPRFIX, JCMDS, JGMDS, and OGMDS, which your application can use to access the message data. DSECT RECPRFIX describes the record prefix, as shown in Table 24 on page 77 and in the "VSE/POWER Record Prefix Layout" in “Spool-Access Support Parameter List (PWRSPL DSECT)” on page 231. The second byte of the prefix indicates that the succeeding record is a fixed format job completion, job generation, or output generation message and equates to RECTFJCM (X'09'), RECTFJGM (X'0A'), and

RECTFOGM (X'0B'). The layout of the fixed format messages is provided by DSECT JCMDS, JGMDS, and OGMDS. For the layout of DSECTs JCMDS (for job completion messages), JGMDS (for job generation messages), and OGMDS (for output generation messages) refer to "Spool-Access Support Parameter List (PWRSPL DSECT)" on page 231.

Since the length of DSECT JCMDS/JGMDS/OGMDS may change in a future release of VSE/POWER, use field RECLNGTH to determine the message length.

For layout and description of a fixed format extended event message (XEM), refer to "GCM-XEM Service" on page 155 below in this chapter.

Message Selection Criteria

When specifying selection criteria for retrieving from a message queue the fixed format job event (job generation and job completion) and output generation messages, you have the following options:

- Retrieve all messages.
- Retrieve all messages resulting from jobs with a specific jobname.
- Retrieve all messages resulting from jobs with a specific jobname and jobnumber.
- Retrieve all messages of a specific type.
- Retrieve all messages of a specific type resulting from jobs with a specific jobname.
- Retrieve all messages of a specific type resulting from jobs with a specific jobname and jobnumber.

To retrieve all messages resulting from jobs submitted by an application, specify:

```
PWRSPL TYPE=UPD,REQ=GCM,USERID=userid
along with:
jobname = '␢␢␢␢␢␢␢␢'     (field SPLGJB)
jobnumber = X'0000'      (field SPLGJN)
sub-request = X'00'      (field SPLGSRB)
```

To retrieve all messages resulting from jobs identified by jobname, specify

```
PWRSPL TYPE=UPD,REQ=GCM,JOBN=jobname,USERID=userid
along with:
jobnumber = X'0000'      (field SPLGJN)
sub-request = X'00'      (field SPLGSRB)
```

To retrieve all messages resulting from jobs identified by jobname and jobnumber, specify

```
PWRSPL TYPE=UPD,REQ=GCM,JOBN=jobname,JNUM=fieldname,USERID=userid
with:
sub-request = X'00'      (field SPLGSRB)
```

To retrieve all messages of a specific type resulting from jobs submitted by an application, specify:

```
PWRSPL TYPE=UPD,REQ=GCM,USERID=userid
along with:
jobname = '␢␢␢␢␢␢␢␢'     (field SPLGJB)
jobnumber = X'0000'      (field SPLGJN)
sub-request = SPLGSRJG to retrive job generation messages
            = SPLGSRJC to retrive job completion messages
            = SPLGSROG to retrieve output generation messages
              (field SPLGSRB)
```

To retrieve all messages of a specific type resulting from jobs identified by jobname, specify:

```
PWRSPL TYPE=UPD,REQ=GCM,JOBN=jobname,USERID=userid
along with:
jobnumber = X'0000'      (field SPLGJN)
sub-request = SPLGSRJG to retrieve job generation messages
            = SPLGSRJC to retrieve job completion messages
            = SPLGSROG to retrieve output generation messages
              (field SPLGSRB)
```

To retrieve all messages of a specific type resulting from jobs identified by jobname and jobnumber, specify:

```
PWRSPL TYPE=UPD,REQ=GCM,JOBN=jobname,JNUM=fieldname,USERID=userid
along with:
sub-request = SPLGSRJG to retrieve job generation messages
            = SPLGSRJC to retrieve job completion messages
            = SPLGSROG to retrieve output generation messages
              (field SPLGSRB)
```

It is also possible to use TYPE=GEN, if you want to use a new SPL. If you omit the JOBN operand, VSE/POWER uses 'AUTONAME' as the default for jobname.

GCM-OPEN Request Types

The following GCM-OPEN requests are available to access a message queue:

- GCM-OPEN-DELETE request
- GCM-OPEN-KEEP request
- GCM-OPEN-REMOVE request
- GCM-OPEN-PURGE request

The table below shows the SPL fields which are applicable to the various GCM request types:

Table 46. SPL Fields Applicable to a GCM-OPEN Request

Field Name	Applicability	Purpose/Contents
SPLGUS	M	User ID[1, 2]
SPLGJB	O	Job name[1, 3]
SPLGJN	O	Job number[1, 3]
SPLGFB1	M	Function byte 1
SPLGSRB	O	Sub-request byte[3]
SPLXWAIT	O	Wait interval
SPLGOPT2.SPLGO2CD	O	Alternative selection criteria
SPLGOPT2.SPLGO2WP	O	Is used to enable an additional GCM-OPEN WAIT request during VSE/POWER shutdown (only applicable for GCM-OPEN-KEEP or GCM-OPEN-DELETE)
SPLGOPT2.SPLGO2OJ	O	Check with original job number
Legend: M = Mandatory; O = Optional		
[1] Generally defined in the PWRSPL macro along with other spool control values. [2] Part of the message queue address. [3] Used to define the selection criteria.		

Issuing a GCM-OPEN-DELETE Request

You issue a GCM-OPEN-DELETE request by coding a GCM-OPEN request with function byte SPLGFB1 equated to SPLGF1DM.

This request retrieves job event and output generation messages, then stores them in your reply buffer, and deletes them from the queue at the time of retrieval. For example, you apply it, when you are sure that you do not need to retrieve again the already processed messages. Generally, this is the case, when it is your application program which validates the messages.

Since all retrieved messages are removed from the queue at the point of retrieval, there is a lesser risk of running into message queue space shortage as with the GCM-OPEN-KEEP request.

For the return and feedback information, see Table 49 on page 165 and "Issuing a GCM-OPEN-KEEP Request."

Issuing a GCM-OPEN-KEEP Request

You issue a GCM-OPEN-KEEP request by coding a GCM-OPEN request with function byte SPLGFB1 equated to SPLGF1KM

This service keeps the job event and output generation messages at the time of retrieval in the queue and stores them in your reply buffer. This allows your program to retrieve messages again if needed.

In some cases it may be required that your application send the retrieved messages to another program or to a remote application. It may happen that the transmission is disrupted and the messages are lost. In such cases, your application is able to obtain already retrieved messages again, provided that a GCM-OPEN-KEEP request has been used for the relevant messages.

Finally, when all of your messages are successfully submitted to the remote application, your program should issue a GCM-REMOVE subrequest in order to delete all messages which match the selection criteria you specified in the GCM-OPEN-KEEP request and which your application has already retrieved.

You may provide code in your program to handle some situations which are reflected by the following return/feedback combinations in fields PXPRETCD/PXPFBKCD. Your code may also inspect the contents of field IJBXSLN, which reflects the actual length of data sent to your program contained in your reply buffer. For the GCM-OPEN-KEEP and GCM-OPEN-DELETE the return and feedback codes may be as follows:

PXPRCOK/PXP00OK (X'00'/X'00') and field IJBXSLN > 0.
: Job event and output generation messages are contained in your reply buffer, but there are more messages to retrieve by your program.

PXPRCOK/PXP00EOD (X'00'/X'01') and field IJBXSLN ≥ 0.
: Job event and output generation messages are contained in your reply buffer. All messages retrieved.

PXPRCOKF/PXP04NJC (X'04'/X'12')
: Job event and output generation message retrieval not available for the GCM-OPEN request, because the message queue size has been defined with JCMQ=0 during the VSE/POWER startup.

PXPRCOKF/PXP04NMF (X'04'/X'16')
No job event or output generation message has been found for the GCM-OPEN request.

PXPRCERR/PXP08BTS (X'08'/X'1A')
The reply buffer which you have defined is too small to contain at least one fixed format job event or output generation message and a prefix.

For additional return and feedback information, see Table 49 on page 165.

Issuing a GCM-OPEN-REMOVE Request

You issue a GCM-OPEN-REMOVE request by coding a GCM-OPEN request with function byte SPLGFB1 equated to SPLGF1RM.

This service deletes queued messages without passing them to your program. You may code this request if you want to purge the message queue from all messages which match the selection criteria you pass along with this request and which have already been retrieved.

For the GCM-OPEN-REMOVE request, the return and feedback code may be:

PXPRCOK/PXP00OK (X'00'/X'00')
IJBXSLN=0: request processed and all messages deleted.

PXPRCOKF/PXP04NMF (X'04'/X'16')
No message found to delete.

For additional return and feedback information, see Table 49 on page 165.

Issuing A GCM-OPEN-PURGE Request

A GCM-OPEN-PURGE request is issued by coding a GCM-OPEN request with function byte SPLGFB1 equated to SPLGF1PM.

This service is comparable to the GCM-OPEN-REMOVE request, but removes queued messages regardless whether they have been already retrieved or not. All selection criteria valid for the GCM-OPEN-REMOVE request can also be specified for the GCM-OPEN-PURGE request.

One single request can also be used to delete messages of one or more queues of the same XPCC-applid. To delete messages for

- A specific user - field SPLGUS must specify the user
- All users - field SPLGUS must contain 8 hexadecimal blanks (X'40'). The value cannot be specified for other GCM and Spool-access support requests.
- A common user - field SPLGUS must contain eight X'FF' characters

The request will only operate on job event or output generation message queues identified by the actual XPCC-applid.

For the GCM-OPEN-PURGE request the return and feedback codes may be:

PXPRCOK/PXP00OK
Request processed and all messages deleted. IJBXSLN is set to zero.

PXPRCOKF/PXP04NMF
No message found to delete

For additional return and feedback information, see Table 49 on page 165.

Optional Specifications Related to the GCM-OPEN Request

Several options can be specified along with a GCM-OPEN request. These options are specified in the following SPL's fields:

1. **SPLXWAIT.** Is used to specify a WAIT interval.
2. **SPLGOPT2.** Is used to enable an additional GCM-OPEN WAIT request during VSE/POWER shutdown.

SPLXWAIT

Usually, the GCM-OPEN-DELETE/KEEP/REMOVE/PURGE requests can be referred to as the 'immediate' requests, because VSE/POWER executes the desired action immediately and at the same time completes the user's pending XPCC SENDR request with or without job event or output generation messages being accompanied by the corresponding RC/FDBKs.

GCM-OPEN-DELETE/KEEP requests can also be used with a WAIT option specified in the SPL.

SPLXWAIT times must be specified in units of seconds. They are interpreted as follows:

X'0000'
: (default), no wait time specified. The request is handled as immediate GCM-OPEN-DELETE/KEEP

X'0001'-X'FFFF'
: valid wait time specified. The request will wait at most that many seconds for a message to be queued. The maximum value which VSE/POWER accepts is at most decimal 27962 or hexadecimal X'6D3A'. Any value above will be accepted but handled by VSE/POWER as 'indefinite' wait specification without warning the user.

The application program's XPCC SENDR request is posted complete by either

- A selectable message(s), before the wait interval has expired
- Or, by a 'nothing found', RC/FDBK=PXPRCOKF/PXP04NMF (=X'04'/X'16') condition, when the time interval has expired, and just before a selectable job event and output generation message has been queued.

When the XPCC reply buffer is too small to hold all selectable messages, the existing GCM-MORE subrequest may be used to retrieve further messages. This subrequest is handled as 'immediate'. Only with a new GCM-OPEN-DELETE/KEEP request a new wait time interval can be specified and will be honored.

For GCM-OPEN-REMOVE and GCM-OPEN-PURGE (see below) requests any time specification in field SPLXWAIT is ignored.

Multiple GCM-OPEN WAIT requests, that is requests with the same XPCC applid and Spool-Access support SPLGUS userid, are not allowed. VSE/POWER reflects such a situation with return and feedback codes (PXPRCNOK/PXP10CAA) and terminates the connection.

Note: If a GCM-OPEN-KEEP WAIT request follows immediately a preceding GCM-OPEN-KEEP WAIT request, the request may complete immediately, because already retrieved messages are still available for the next GCM retrieval request.

SPLGOPT2

Using selection criteria of GCM-OPEN request, you can retrieve the messages resulting from jobs that have specific job name, job number, or both, as described in "Message Selection Criteria" on page 144. If generating job name, job number, or both specified in the selection criteria match those job name, job number, or both that are kept in a message, then this message will be retrieved. Additional options in the option byte SPLGOPT2 of SPL provide extended selection criterion as follows:

- SPLGO2OJ

  This option can be used with JCM, JGM and OGM, which are generated by jobs submitted with the same option via PUT-OPEN request (see "Additional Job Event and Output Generation Message Options" on page 110). If this option is specified, then the original generating job number is used as selection criteria. Original generating job number is the one that is taken on the system where a job was initially submitted. Remember that after submission on a system, the job can be transferred by PNET for processing to another system where it gets a new number.

  If option SPLGO2OJ is used, then the job number passed along with the GCM-OPEN request will be compared with the job number that is kept in the field JGMF1NUM of JGM, JCMFONUM of JCM, or OGMFONUM of OGM.

- SPLGO2CD

  This option affects generation messages only (JGM and OGM) and is ignored for completion message (JCM). SPLGO2CD option overrides SPLGO2OJ option if both are specified.

  If SPLGO2CD is specified, then the job name and number of a generated job (for JGM) or the job name and number of generated output (for OGM) are used as selection criteria. Due to option SPLGO2CD, the job name and number passed along with GCM-OPEN request will be compared with the job name and job number that are kept in the fields JGMFNNAM and JGMFNNUM for JGM, or in the fields OGMFNNAM and OGMFNNUM for OGM. For example, this option can be used to determine whether a specific job, whose name, number, or both are known, has already been created.

- SPLGO2OJ and SPLGO2CD not specified

  In this case actual generating job number is used as selection criteria. Actual generating job number is the one that a job gets on the system where it is actually processed. If additional options are omitted, the job name and job number passed along with GCM-OPEN request will be compared with the job name and job number which are kept in the fields JCMFNAM and JCMFNUM (of a job completion message), JGMFNAM and JGMFNUM (of a job generation message), and OGMFNAM and OGMFNUM (of an output generation message), refer to "Spool-Access Support Parameter List (PWRSPL DSECT)" on page 231. This is the default selection criteria.

You can enable an additional last GCM-OPEN WAIT request within a single XPCC connection during VSE/POWER shutdown. Use the additional option in the option byte SPLGOPT2 of SPL as follows:

- SPLGO2WP. By default, VSE/POWER terminates outstanding GCM-OPEN WAIT requests during PEND processing. However, if you need to retrieve any essential messages before VSE/POWER shutdown is complete, you can specify SPLGO2WP option to force VSE/POWER to accept an additional last GCM-OPEN WAIT request during shutdown. Specify only a finite wait interval in the further request to prevent unnecessary delay in the VSE/POWER termination.

The SPLGO2WP option doesn't affect PEND FORCE and PEND IMM processing, GCM sub-requests (GCM-MORE and GCM-REMOVE) ignore this option.

Retrieving Messages from Common Queues

An application program that wants to address a common message queue by the GCM Service, must connect to VSE/POWER with the corresponding XPCC-applid and must specify the X'FF...FF' common userid in SPLGUS to match the identification of the desired queue. All GCM Services valid for the default single job event and output generation message queues may also be used to request messages from common queues. The fields ...FPRIV and ...FUSID of the dummy sections JCMDS, JGMDS, and OGMDS may be helpful in identifying the owner of a message when the message is retrieved from a common queue.

GCM Subrequests

Depending on the request type used and the accompanying return and feedback information, it may be desirable to continue with a GCM-MORE subrequest or a GCM-REMOVE subrequest. Refer to "Coding Sequence for a GCM Service" on page 153 for the coding sequence allowed for these two subrequest types.

Issuing a GCM-MORE Subrequest

Your application should issue a GCM-MORE subrequest when it is informed that not all messages fit in its reply buffer. This is indicated by the return and feedback code combination PXPRETCD/PXPFBKCD=X'00'/X'00' and any data length indicated in the IJBXSLN field. Your program should repeat the GCM-MORE subrequest so long as there are still messages to retrieve. For a GCM-MORE subrequest, do the following coding:

- Set byte PXUBTYP of the XPCCB to zero.
- Set byte PXUACT1 of the XPCCB to the value equated to PXUATGCM
- Issue an XPCC FUNC=SENDR request passing a null buffer (set IJBXBLN to zero).

For the return and feedback codes provided by VSE/POWER, please refer to the GCM-OPEN-KEEP or GCM-OPEN-DELETE requests and Table 49 on page 165.

Issuing a GCM-REMOVE Subrequest

Your program should issue a GCM-REMOVE subrequest when you are sure that all messages have reached your remote application. This request deletes all messages which match the selection criteria of the actual GCM-OPEN request and which have been retrieved already. You can only issue this request after a GCM-OPEN-KEEP request. For a GCM-REMOVE subrequest, do the following coding:

- Set byte PXUBTYP of the XPCCB to zero.
- Set byte PXUACT1 of the of the XPCCB to the value equated to PXUATDEL
- Issue an XPCC FUNC=SENDR request passing a null buffer (set field IJBXBLN to zero).

VSE/POWER informs you with the return and feedback information PXPRCOK/PXP00EOD (X'00'/X'01') if all messages have been deleted.

Additional Considerations

Wait Specification

When specifying the wait option, most of the interactions needed to retrieve messages are equal to those needed for the immediate GCM request. However, some considerations about terminating a 'waiting for message' request should be done.

The following events will terminate a waiting request:

Message event
The application program's XPCC SENDR ECB is posted with one or more messages available in the reply buffer

Wait completion
The application program's XPCC SENDR ECB is posted with RC/FDBK=PXPRCOKF/PXP04NMF and an empty reply buffer

PSTOP SAS cmd.
The application program's XPCC SENDR ECB is posted with XPCC reason code IJBXDISC and IJBXCPRG.

Program action
The application program's logic decides to stop waiting by an XPCC DISCPRG request.

PEND command
Gradual termination of all active VSE/POWER jobs on job boundary is desired. Therefore a waiting application should be informed that VSE/POWER has entered the termination (PEND) period, where all tasks should cease processing.

1. If VSE/POWER enters the PEND state, while a GCM-OPEN WAIT request is in progress, the application is posted immediately and PXPIPSH is passed on to the application in information byte PXPINFO. The connection is terminated.
2. If the GCM-OPEN WAIT request has been set up with SPLGO2WP, the application is posted immediately, and PXPIPSH is passed on to the application. The application may issue a GCM-OPEN WAIT request with SPLGO2WP again. Any other request will be rejected.

Special Userid

Field SPLGUS used in the GCM-OPEN-DELETE/KEEP/REMOVE/PURGE requests may contain the 8-byte hexadecimal value X'FF...FF'. This value is used to identify a common job event and output generation message queue.

The value cannot be specified for PUT,GET and CTL Spool-access requests.

Reflecting Common Job Event and Output Generation Message Queues

For better operator readability a common job event and output generation message queue is identified by the 8-byte identifier '-COMMON-' replacing the USERID placeholder in the

- Statistics Status Report (Support for Retrieval of Job Event and Output Generation Messages)
- Message 1Q4AI - to reflect the loss of messages

- Message 1R48I - the PDISPLAY Active report

Identifying The Lost Message Condition

While job event and output generation messages are queued to a job event message queue, it may happen that a message is dropped because the specification in the SET JCMQ statement was too small. The application which retrieves messages is informed by field PXPLEMC, located in the user data as an overlay of field PXPROFF, about the number of lost messages at this queue since the last GCM-OPEN request completed. The lost message count is passed to the application only for GCM-OPEN WAIT requests.

Additionally, message 1Q4AI is issued on the console in a time interval of 60 seconds whenever a job event or output generation message is discarded.

The statistics status report identifies the XPCC application ID and PWRSPL userid of the job event and output generation message queue which has the largest amount of lost messages of all existing queues since the last VSE/POWER startup.

Reflecting Active GCM Applications

Any immediate GCM request being handled by VSE/POWER can be made visible by the PDISPLAY A,SAS command, which in turn identifies the corresponding VSE/POWER service task by the existing console display message 1R48I. A new waiting GCM request may be visible for a long period. It must even be addressable by the central operator. Therefore, message 1R48I is extended to identify Spool-access support connections, which wait for job event and output generation messages:

```
1R48I  SAS,conn-id, SAS=xpcc-applid,splgus-userid, REQ=GCM
1R48I  SAS,conn-id, SAS=xpcc-applid,splgus-userid, REQ=GCM-WAIT
```

If a common message queue is affected, *splgus-userid* is replaced by the 8-byte constant '-COMMON-'.

A 'blank' splgus-userid used in the GCM-OPEN-PURGE request will be displayed by the 8-byte constant '--BLNK--'.

Multiple GCM Requests

In order to retrieve fixed format job event and output generation messages it may be necessary to code several application programs and run them at the same time, or it may be necessary to code a sequence of GCM-OPEN requests in one single program. In such cases, the following should be taken into consideration.

- Concurrent GCM Requests

  If several application programs process GCM requests at the same time, it is recommended that each program use its own specific pair of XPCC application-ID and PWRSPL user-ID. This ensures that no messages are retrieved and deleted by an application program while these messages are expected by another application program.

  For example, consider two programs which use the same application ID and user ID along with their GCM requests. Furthermore, assume that program A issues a GCM-OPEN-KEEP request. After retrieval of the first message buffer, VSE/POWER signals that there are more messages to retrieve. At this point it may happen that program B gets control and issues a GCM-OPEN-DELETE request. It is now very likely, that messages are retrieved and deleted by program B which are expected to be retrieved by the next GCM-MORE request of program A. If this occurs, it is possible that not all messages or even no message at all be retrieved by the GCM-MORE request.

- Sequential GCM Requests

  If you code several GCM-OPEN requests in one application program and you want to use the same XPCC application ID and PWRSPL userid with the next sequential GCM-OPEN request, it is a good practice to finish a GCM-OPEN-KEEP request first by issuing a GCM-REMOVE or GCM-OPEN-REMOVE request. However, if you do not purge the message queue before your next GCM-OPEN request starts processing, you will retrieve all messages which match the specified selection criteria, regardless of whether these messages have been previously retrieved or not.

Shared Processing

VSE/POWER will return the job event and output generation messages to that system of a shared system complex on which the original job has been submitted. If, for example, a job which has been submitted with option SPLGF1QM on the system with the VSE/POWER SYSID=3, is processed on another system (SYSID≠3), VSE/POWER will ensure that:

- The event message of this job is returned to the originating SYSID=3 where
- The event message is queued to the message queue specified by the job submitter. If, however, a disk I/O error occurs while returning messages, the messages may be lost.

Networking

If a job is submitted with the 'queue-event-message' option and this job is processed on another node (which must be a VSE/POWER node of at least Version 5.2), the resulting job event and output generation messages are returned to the originating node and queued for retrieval.

Discontinuing the GCM-Service

You can terminate the GCM service using either of these ways:

1. Request another OPEN request (GET, PUT, CTL, GCM) of the spool-access support.
2. Specify a DISCONN or DISCPRG XPCC request.

Coding Sequence for a GCM Service

The following coding sequence shows the steps in which the spool-access support user's application program interacts with VSE/POWER using the GCM-OPEN-KEEP or GCM-OPEN-DELETE service.

Table 47. GCM Service Processing Sequence

Step	Coding in your application program	Comments
	... ...	
1	Open the service XPCC FUNC=SENDR	Your program's send buffer must contain an SPL generated for requesting the GCM service for XPCC-applid.userid.

Table 47. GCM Service Processing Sequence (continued)

Step	Coding in your application program	Comments
2	Check the return codes in register 15 and in the XPCCB (byte IJBXRETC)	
3	WAIT IJBXSECB	
4	Check the reason code (in the XPCCB byte IJBXREAS)	
5	Check the VSE/POWER return and feedback codes (in the XPCCB bytes PXPRETCD and PXPFBKCD, respectively)	
6	Check for and evaluate messages returned by VSE/POWER	If messages are to be returned for the XPCC-applid.userid application, then VSE/POWER passes them to your program's reply buffer.
7	If feedback code PXP00EOD does not indicate availability of additional messages, **go to step 9**; else proceed.	
8	Get additional messages by XPCC FUNC=SENDR,... and **return to step 3**.	Coding for this purpose is required only if the feedback code indicates that more messages are queued. Your program must set up a GCM-MORE request.
9	End of Service	

The following coding rules and sequences of the various GCM-OPEN requests and GCM subrequests are valid:

GCM-OPEN-KEEP Subrequests

```
                   ┌──────────────────────┐
                   ▼                      │
►►──GCM-OPEN-KEEP──┴─┬──────────────────┬─┴────────────────────────────►◄
                     ├──GCM-MORE────────┤
                     └──GCM-REMOVE──────┘
```

GCM-OPEN-DELETE Subrequest

```
                     ┌──────────────────┐
                     ▼                  │
►►──GCM-OPEN-DELETE──┴─┬──────────────┬─┴────────────────────────────►◄
                       └──GCM-MORE────┘
```

```
►►──GCM-OPEN-REMOVE──────────────────────────────────────────────────►◄
```

```
►►──GCM-OPEN-PURGE───────────────────────────────────────────────────►◄
```

If the sequence rules are not obeyed, VSE/POWER stops the request, and PXPRETCD/PXPFBKCD = PXPRCERR/PXP08ROS (X'08'/X'25) is returned to the application.

GCM-XEM Service

GCM-XEM is an extension of the GCM service for eXtended Event Message support. A user-written application program uses this extended service for both initializing generation of extended event messages and extracting queued messages.

Overview of eXtended Event Messages Handling

The generation of XEMs differs significantly from the generation of JCMs/JGMs/OGMs. JCMs/JGMs/OGMs are created by VSE/POWER only for a job which was submitted via the SAS interface with specific options. XEMs are not related to the execution of any specific job. An application program initializes XEM using a special SAS request. Subsequently, VSE/POWER begins to generate XEMs until the application program terminates this process.

XEMs cover a much wider set of events compared to JCMs/JGMs/OGMs (refer to “Generation of eXtended Event Messages” on page 156). Therefore, while only a few JCMs/JGMs/OGMs are issued, many XEMs can be produced.

VSE/POWER puts a generated XEM in a separate message queue which is not used for keeping JCMs/JGMs/OGMs. This queue has a fixed number of message slots (one slot for one message). If there is no free slot for a new message (the queue is full), then the message is discarded and lost for the application.

VSE/POWER provides every XEM application with its own queue; the total number of concurrent XEM applications is limited by a predefined value (refer to “XEM Support Capacity” on page 157). A message queue is only accessible by the initiating application. Thus, the same message can be kept within several queues owned by different applications simultaneously. Retrieving a message from one application queue doesn't result in retrieving the same message from another application queue.

At initialization of the XEM service, an application program can specify selection criterion – types of queue entries whose events will result in messages queuing. RDR, LST, or PUN entries can be selected here, as well as any combinations of these types.

VSE/POWER returns queued messages to an application as batches, that is, several messages at once within a reply buffer. Similar to JCMs/JGMs/OGMs, each XEM is preceded in the reply buffer by a record prefix. The reply buffer has a fixed length (refer to “Retrieving eXtended Event Messages” on page 160), so an application must reserve storage of the predefined size for this buffer. Note, however, that the buffer may not completely filled if fewer messages are available at retrieval time.

Messages are returned by VSE/POWER in FIFO order. There is no selection criteria for messages retrieval (all queued messages are returned unconditionally). The slot of a retrieved message becomes available for storing a new message.

Concurrent usage of the GCM-XEM service by several applications with the same ID is not allowed. VSE/POWER rejects attempts to repeatedly start the XEM service by the same application ID.

Generation of eXtended Event Messages

VSE/POWER generates XEMs when the following events occur:

- A new entry has been created within a VSE/POWER queue or spooled to a tape. An entry can be created, for example, by spooling or segmenting output, submitting a job via reader or SAS application, punching the account file, duplicating another entry (by using * $$ LSTDUP, or * $$ PUNDUP statement, or PCOPY operator command).
- An existing entry has been altered in a VSE/POWER queue. An entry is altered as a result of processing (with initial disposition K) or issuance of specific operator commands. Entry processing here means, for example, printing or punching an output, getting an entry via SAS interface (or by GETSPOOL macro) or sending it via PNET. Operator commands that can alter queue entries are: PRELEASE, PHOLD and PCANCEL, externally or internally invoked PALTER and PFLUSH. Note, that browsing an entry doesn't result in its alteration and, therefore, is not recorded.
- An existing entry has been deleted from RDR, LST, PUN or XMT queue. Deletion takes place, for example, if an output (with initial disposition D) is printed or punched, a job (with DISP=D) is executed or canceled (PCANCEL operator command issued), an entry is deleted by PDELETE command, or the expiration time of an entry is reached (PDELETE is invoked internally).

Destination of eXtended Event Messages

VSE/POWER uses the application ID specified via the APPL operand of the XPCCB macro to address the message queue for keeping extended event messages. VSE/POWER ignores the user ID specified in the USERID operand of the PWRSPL macro, thus an application program can omit this specification for GCM-XEM requests. The output of the 'PDISPLAY A,SAS' operator command always shows *XEM* as the user ID for status of the SAS task which has started the XEM service. Refer to *VSE/POWER Administration and Operation*, SC34-2625 for more detailed information.

VSE/POWER reserves storage for the application XEM queue when the application starts the XEM service, and releases the storage when the application stops the service (or disconnects VSE/POWER). While the queue exists, newly generated extended event messages are saved in it, and the pertinent application can retrieve them from the queue. VSE/POWER actually processes messages in the following way. When a new extended event message has been produced (a queue entry was created, altered or deleted), VSE/POWER looks through the running applications which have started an XEM service and puts the message into those queues where a specified selection criterion is satisfied. After that, it starts waiting for a new message.

When storage of any application message queue is released, all messages that were not retrieved are discarded for this application.

Storage Allocation for XEM Support

Storage for XEM support is reserved in the two-step procedure as follows:

- VSE/POWER on its own startup reserves storage for the XEM Control Block (XMCB) which controls addresses of extended event message queues. The lifetime of XMCB continues until VSE/POWER shutdown.
- When an application starts the XEM service, VSE/POWER reserves storage for the message queue of this application. This storage is released when the application stops the XEM service.

The size of the XEM Control Block (XMCB) does not exceed 4 KB of real (fixed) storage of the VSE/POWER partition. One message queue occupies 512 KB within the GETVIS-31 area of the VSE/POWER partition. Ensure that sufficient real storage is available for the XMCB, and sufficient GETVIS-31 storage is available for the applications' message queues. If there is no sufficient real storage for the XMCB, then XEM support is unavailable. If GETVIS-31 storage is insufficient for an application message queue, then VSE/POWER cannot start the XEM service for the requesting application.

VSE/POWER does not provide any notification about insufficient real storage for the XMCB during its startup. Instead, it informs an application program and operator as follows:

- Return and feedback codes are loaded into the verification SPL which is returned to the application after the GCM-XEM-START request (refer to “Starting the GCM-XEM Service” on page 159).
- Message 1Q3KI (RC=0001) is displayed on the system console after the GCM-XEM-START request has been issued.
- The output of the ‘PDISPLAY STATUS’ operator command shows that XEM support is unavailable.

If VSE/POWER cannot start the XEM service for an application because of insufficient GETVIS-31 storage, it notifies the requesting application program and operator as follows:

- Return and feedback codes are loaded into the verification SPL and returned to the application after the GCM-XEM-START request (refer to “Starting the GCM-XEM Service” on page 159).
- Message 1Q3KI (RC=0004) is displayed on the system console after the GCM-XEM-START request has been issued.

XEM Support Capacity

Regarding storage requirement for XEM support, refer to “Storage Allocation for XEM Support.”

The message queue of an application contains 2048 slots for keeping extended event messages (one slot for one message). This number is reflected in the output of the ‘PDISPLAY STATUS’ operator command. If there is no free slot for a new message (the queue is full), then the message is discarded (lost for the application program) and VSE/POWER notifies the application and operator as follows:

- Number of lost messages is returned to the application during messages retrieval in the two-byte field PXPLEMC of the XPCCB (see “Retrieving eXtended Event Messages” on page 160).

- Message 1Q4AI RC=0004 is displayed on the system console each time an extended event message is lost for any application and when at least 60 seconds have passed since the last appearance of 1Q4AI RC=0004 (see "Retrieving eXtended Event Messages" on page 160).
- Output of the 'PDISPLAY STATUS' command shows ID of the application which lost the maximal number of extended event messages and this number itself.

VSE/POWER supports up to 32 running applications which use XEM service concurrently. This number is reflected in the 'PDISPLAY STATUS' command output. If the limit is exceeded, then VSE/POWER informs the requesting application program and operator as follows:
- Return and feedback codes are passed along with the verification SPL which is returned to the user application program after the GCM-XEM-START request (see "Starting the GCM-XEM Service" on page 159).
- Message 1Q3KI (RC=0002) is shown on the system console after the GCM-XEM-START request has been issued.
- Report of the 'PDISPLAY STATUS' command shows number of currently running applications which use XEM support.

How to Use XEM Support

XEM support is provided with the extended GCM service only (GCM-XEM). This means that an application invokes this extended service for both initializing message queuing and retrieving queued messages. Such behavior differs from usage of JCM/JGM/OGM support, when an application requests PUT (or CTL) service to submit a job and to initialize messages queuing, and later uses GCM service for retrieving messages.

In general, after an application has been identified to XPCC and connection to VSE/POWER has been established (as described under "Setting Up a Communication Path" on page 59), you must adhere to the following steps for inquiring extended event messages:
- Start GCM-XEM service (get access to XEM support).

  When an application has requested the service, VSE/POWER reserves storage for the application message queue and begins queuing messages for this application. Start of XEM service doesn't itself result in retrieving messages. To retrieve queued messages, the application must open GCM-XEM service.
- Retrieve XEM message.

  To retrieve messages, the application must obtain access to the XEM message queue (in other words, open GCM-XEM service). When the service is opened by the application, VSE/POWER waits until the reply buffer of this application is full. When the buffer is full or the waiting period (explicitly specified by the application or the default) is expired, then the reply buffer is returned to the application. If all available messages do not fit in the reply buffer (which is indicated by return and feedback codes), then the application can retrieve them immediately using GCM-MORE sub-request. An application can also reopen the GCM-XEM service to retrieve messages that were not returned and to wait for new messages.
- Stop GCM-XEM service (close access to XEM support).

  VSE/POWER releases storage occupied by the application message queue (all messages that were not retrieved are discarded).

For details, refer to the sections below.

Layout of a Fixed Format eXtended Event Message

Layout of an extended event message data record within the reply buffer is similar to layout of the other fixed format messages (refer to "Layout of a Fixed Format Job Event and Output Generation Message" on page 143):

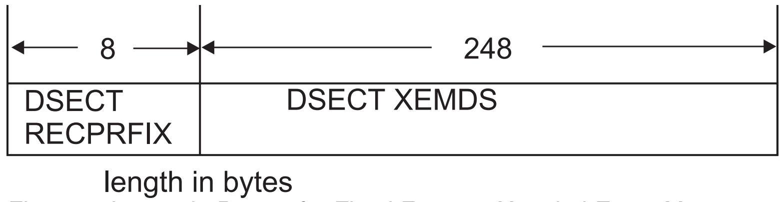


Figure 8. Layout in Bytes of a Fixed Format eXtended Event Message

The PWRSPL macro provides DSECT's RECPRFIX and XEMDS which applications can use to access the extended event message data. For a description of the record prefix (DSECT RECPRFIX), refer to Table 24 on page 77; and to the "VSE/POWER Record Prefix Layout" section in "Spool-Access Support Parameter List (PWRSPL DSECT)" on page 231. The second byte of the prefix indicates that the following record is a fixed format extended event message and is equal to RECTFXEM (X'0C'). For layout of a fixed format extended event message (DSECT XEMDS), refer to "Spool-Access Support Parameter List (PWRSPL DSECT)" on page 231.

Since the length of DSECT XEMDS may change in a future release of VSE/POWER, use field RECLNGTH to find out the actual length of an extended event message.

Starting the GCM-XEM Service

To start the XEM service, an application program must issue the GCM-XEM-START request with the following specifications:

- Specify SEND buffer type within XPCCB as SPL: set field PXUBTYP equated to PXUBTSPL, specify SPL address and length in the XPCCB fields IJBXADR and IJBXBLN.
- Set up an SPL as

  ```
  PWRSPL TYPE=UPD,REQ=GCM
  ```

  and mandatory function byte SPLGFB1 equated to SPLGF1XS.

  Instead of TYPE=UPD, TYPE=GEN can be used, if you want to specify a new SPL. As opposed to the GCM-OPEN request, specification of USERID is not needed for GCM-XEM-START, and actually VSE/POWER ignores this specification (refer to "Destination of eXtended Event Messages" on page 156).

  Optionally, you can reduce the stream of queuing messages by specifying selection criterion within the extended flag byte SPLXFLG1. Specify:

  – SPLX1XRD to queue event messages related to RDR entry type only;

  – SPLX1XLS to queue event messages related to LST entry type only;

  – SPLX1XPN to queue event messages related to PUN entry type only.

  To select more than one queue entry type, apply the logical sum of the above flags. If you omit specification of selection criterion, VSE/POWER will queue extended event messages for entries of all types (RDR, LST, and PUN).

  Note: Selection criterion determines queue entry types, but not VSE/POWER queues. If, for example, you specify SPLX1XLS, it will result in queuing event messages for both LST and XMT (I=L) entries. The actual location of an entry is flagged within the message; refer to DSECT XEMDS in "Spool-Access Support Parameter List (PWRSPL DSECT)" on page 231.

VSE/POWER indicates the result of GCM-XEM-START request processing by means of return/feedback codes combination in fields PXPRETCD/PXPFBKCD and, additionally, by feedback-2 code in the field PXPFBKC2. After the application program has been posted in the field IJBXSECB of XPCCB, it can analyze these codes:

PXPRCOK/ PXP00OK (X'00'/X'00')
VSE/POWER has started XEM service for requesting application successfully.

PXPRCERR / PXP08XUA / PXPC24CA (X'08'/X'4C'/X'01')
XEM support is unavailable because there is insufficient real (fixed) storage for the XMCB. These codes are accompanied by 1Q3KI console message with RC=0001 (refer to "Storage Allocation for XEM Support" on page 157).

PXPRCERR / PXP08XUA / PXPC24CB (X'08'/X'4C'/X'02')
VSE/POWER can not start XEM service for an application because the maximum number of applications which can use XEM support concurrently would be exceeded. Codes are accompanied by 1Q3KI console message with RC=0002 (refer to "XEM Support Capacity" on page 157).

PXPRCERR / PXP08XUA / PXPC24CC (X'08'/X'4C'/X'03')
VSE/POWER can not start XEM service for an application because XEM application with the same ID is already running. These codes are accompanied by 1Q3KI console message with RC=0003.

PXPRCERR / PXP08XUA / PXPC24CC (X'08'/X'4C'/X'04')
VSE/POWER can not start XEM service for an application because there is insufficient GETVIS-31 storage for the application message queue. These codes are accompanied by 1Q3KI console message with RC=0004 (refer to "XEM Support Capacity" on page 157).

For additional return and feedback information, see Table 49 on page 165.

Retrieving eXtended Event Messages

Retrieving extended event messages differs from retrieving other fixed format messages. At first, VSE/POWER returns retrieved XEMs in a 4KB reply buffer which can include up to 16 messages. Secondly, VSE/POWER fills the reply buffer over a specified time interval: XEMs are returned to an application if the buffer is full or the time interval has expired. If a application program does not specify a time interval or if zero interval is specified, then the default value of 10 seconds is used.

To retrieve XEMs, an application must issue GCM-XEM-OPEN request with the following specifications:

- Specify SEND buffer type within XPCCB as SPL (set field PXUBTYP equated to PXUBTSPL), specify SPL address and length in the XPCCB fields IJBXADR and IJBXBLN.
- Set up a message reply buffer to which the retrieved messages will be passed:
  - Specify buffer address in the XPCCB field IJBXRADR;
  - Specify buffer size (number of bytes) in the XPCCB field IJBXRLNG.

    Since VSE/POWER returns messages in the 4 KB reply buffer, each application must specify 4096 for the buffer size in the field IJBXRLNG. If an application specifies a smaller buffer, VSE/POWER will reject the request. A larger buffer will be accepted but will not be completely used.

- Set up an SPL as follows:

```
PWRSPL TYPE=UPD,REQ=GCM
```

and the mandatory function byte SPLGFB1 equated to SPLGF1XM (TYPE=GEN can be used instead of TYPE=UPD). In the same way as for GCM-XEM-START, specification of USERID here is ignored by VSE/POWER (refer to "Starting the GCM-XEM Service" on page 159).

Optionally, you can specify a time interval (number of seconds) within the SPLXWAIT field to overwrite the default value of 10 seconds.

When VSE/POWER receives the GCM-XEM-OPEN request, it starts waiting for filling of the message buffer. When the buffer is full or the wait interval has expired, VSE/POWER returns messages (if any) to the application and posts the request in the XPCCB field IJBXSECB. After that, your program can evaluate return and feedback codes in the fields PXPRETCD and PXPFBKCD (and additionally feedback-2 code in the field PXPFBKC2) to clarify, for example, whether messages are available in the reply buffer or an error occurred:

PXPRCOK/PXP00OK (X'00'/X'00')
The reply buffer is full, and there are more messages that were not retrieved.

PXPRCOK/PXP00EOD (X'00'/X'01')
There are messages in the reply buffer, and no more messages were available at retrieval time.

PXPRCOKF/PXP04NMF (X'04'/X'16')
VSE/POWER didn't find any messages during wait interval.

PXPRCERR/PXP08BTS (X'08'/X'1A')
The size of the reply buffer specified by application program is less than the predefined value of 4096 bytes. In this case VSE/POWER cancels all queued messages (if any) and stops XEM service for the application (in the same way as when processing the GCM-XEM-STOP request, refer to "Stopping the GCM-XEM Service" on page 163).

PXPRCERR/PXP08ROS/PXPC225H (X'08'/X'25'/X'08')
Application has issued GCM-XEM-OPEN request prior to successful GCM-XEM-START request.

For other return and feedback codes, refer to Table 49 on page 165.

Your application program can also inspect the following fields of XPCCB returned by VSE/POWER:

- IJBXSLN: actual length of messages sent to the application. 4 KB reply buffer can be filled incompletely if less than 16 messages are available by the expiration of waiting period.
- PXPLEMC: number of discarded messages (lost by your application). This number is counted since the moment when XEM service was started (request GCM-XEM-START is issued) and neither decremented, nor cleared until service stop (GCM-XEM-STOP is issued). Maximum possible value of PXPLEMC field is X'7FFF', which indicates that the number of discarded messages is equal to or greater than 32767.

For retrieving unreturned messages, an application can issue the GCM-MORE subrequest (see "GCM Subrequests" on page 150). This subrequest can be repeated as many times as needed until all available messages have been retrieved. To retrieve more available messages, the application can also issue the GCM-XEM-OPEN request again with the same or another time interval. Note that the GCM-MORE subrequest is posted by VSE/POWER immediately even if the reply buffer is not full (as opposed to the GCM-XEM-OPEN request, which is posted when the buffer is full or the waiting period has expired).

Applicability of Further Requests for Retrieving eXtended Event Messages

The Table 48 reflects the applicability of further requests for XEMs retrieval depending on the return and feedback codes of the previous GCM-XEM-OPEN and GCM-MORE requests.

Table 48. Applicability of further requests for XEM retrieving

Return/Feedback Codes	Further GCM-MORE	Further GCM-XEM-OPEN
PXP00OK/PXP00OK	Applicable	Applicable
PXP00OK/PXP00EOD	Rejected with Return/Feedback/ Feedback-2 codes PXPRCERR/PXP08ROS/PXPC225I (X'08'/X'25'/X'09')	Applicable
PXPRCOKF/ PXP04NMF	Rejected with Return/Feedback/ Feedback-2 codes PXPRCERR/PXP08ROS/PXPC225D (X'08'/X'25'/X'04')	Applicable

Cancelling eXtended Event Messages Retrieving

Sometimes, you might need to cancel retrieving of messages, that is, to revoke a pending GCM-XEM-OPEN request (for example, if the wait interval specified was too long, but there are no XEM events so far). The following options can be used to accomplish this:

- Issue the XPCC CLEAR request from your application program (refer to *z/VSE System Macros Reference*, SC34-2638 and *z/VSE System Macros User's Guide*, SC33-8407).
- Enter 'PCANCEL jobname' or 'PFLUSH partition' (Format 2) command at the operator console.

As a result, VSE/POWER stops the XEM service for the application (refer to "Stopping the GCM-XEM Service" on page 163) and disconnects from the application (if PCANCEL or PFLUSH operator command was entered when job processing is canceled as well). Note that the XPCC CLEAR function can only be used during message retrieval (otherwise, it will be rejected by the XPCC interface with the return code IJBXNREQ, refer to "Stopping the GCM-XEM Service" on page 163).

Please remember that operator command PSTOP format 9 (PSTOP SAS,ALL|*connect_ID*) does not result in the immediate cancellation of XEMs retrieving. Instead, the command only initiates delayed stopping of the SAS XEM task. The task will be stopped when the pending GCM-XEM-OPEN request is posted.

Stopping the GCM-XEM Service

To stop the XEM service, an application must issue the GCM-XEM-STOP request with the following specifications:

- Specify SEND buffer type within XPCCB as SPL: set field PXUBTYP equated to PXUBTSPL, specify SPL address and length in the XPCCB fields IJBXADR and IJBXBLN.
- Set up an SPL as

  ```
  PWRSPL TYPE=UPD,REQ=GCM
  ```

  and the mandatory function byte SPLGFB1 equated to SPLGF1XT.

When XEM is stopped, VSE/POWER terminates queuing messages for the requesting application and releases the storage occupied by the application's message queue (messages that were not returned, if any, are canceled). After that, your application can issue a new SAS request, for example, the GCM-XEM-START to start the XEM service again with the same or another selection criterion (see “Starting the GCM-XEM Service” on page 159).

VSE/POWER indicates the result of GCM-XEM-STOP request processing by means of return/feedback codes combination in fields PXPRETCD/PXPFBKCD and, additionally, by feedback-2 code in the field PXPFBKC2:

PXPRCOK/PXP00OK (X'00'/X'00')
XEM service has been stopped successfully.

PXPRCERR/PXP08ROS/PXPC225H (X'08'/X'25'/X'08')
GCM-XEM-STOP was issued prior to successful execution of GCM-XEM-START request.

XEM service will also be stopped if the application breaks communication path to VSE/POWER via the XPCC DISCONNECT request (instead of using the GCM-XEM-STOP request), or XEMs retrieving has been canceled (see “Cancelling eXtended Event Messages Retrieving” on page 162). Note however, that the DISCONNECT request will result in actual disconnect if the connection is not busy at that moment. Otherwise, in particular, if there is a pending GCM-XEM-OPEN request, the XPCC interface rejects DISCONNECT.

The table below summarizes system's replies on attempts to stop XEM service depending on:

- the action done by the application program or central operator,
- the status of the GCM-XEM-OPEN request.

	Status of the GCM-XEM-OPEN request	
Action done	**Active (IJBXSECB is pending)**	**Not active (IJBXSECB is posted)**
Application issued XPCC CLEAR request	GCM-XEM service for requesting application is stopped; VSE/POWER is disconnected from the requesting application.	The function is rejected by the XPCC interface (IJBXRETC= IJBXNREQ).

Action done	Status of the GCM-XEM-OPEN request	
	Active (IJBXSECB is pending)	**Not active (IJBXSECB is posted)**
Application issued XPCC DISCONNECT request	The function is rejected by the XPCC interface (IJBXRETC=IJBXNDC2).	GCM-XEM service for the requesting application is stopped; VSE/POWER is disconnected from the application.
Application issued GCM-XEM-STOP request	The request is rejected by the XPCC interface (IJBXRETC=IJBXCBSY).	GCM-XEM service for requesting application is stopped; VSE/POWER is not disconnected from the application and SAS user task continues working.
Operator entered PCANCEL or PFLUSH command on the console	GCM-XEM service for the application is stopped; VSE/POWER is disconnected from the application.	GCM-XEM service for the application is stopped; VSE/POWER is disconnected from the application.

Restrictions of XEM Support

XEM support has the following restrictions:

- VSE/POWER does not route generated XEM messages to other systems of a shared spooling complex, nor to other PNET nodes.
- XEMs are generated for master and duplicate queue entries but without indication whether the queue entry is master or duplicate.
- Creation and deletion of an internal queue entry (class=X'FA') is ignored by XEM support. Thus, XEM is not generated for $SPLnnnn LST entry, which VSE/POWER creates temporarily for accumulating display lines of ‘PDISPLAY queue’ command submitted by CTL service request (refer to Chapter 7, “CTL - Passing a Command,” on page 65).
- Deletion of a queue entry ‘in creation’ is ignored by XEM support. For example, if job input is canceled by PFLUSH operator command (Format 1), XEM is not created. XEM will also not be created for output purged via PURGE=nnnn operand of * $$ LST statement.

Return and Feedback Codes from the GCM Requests

The return and feedback codes provided by VSE/POWER for the GCM service requests are described in Table 49. The meaning of these codes is shown in Table 80 on page 297.

Table 49. Return and Feedback Codes for GCM-Service-Related Requests

			Request Type						
Mnemonic	**Return Code**	**Feedback Code**	**GCM-OPEN (KEEP / DELETE)**	**GCM-MORE**	**GCM-REMOVE**	**GCM-OPEN REMOVE / PURGE**	**GCM-XEM-START**	**GCM-XEM-OPEN**	**GCM-XEM-STOP**
PXP00OK	00	00	X	X	X	X	X	X	X
PXP00EOD		01	X	X				X	
PXP04SOA	04	09	X			X			
PXP04NJC		12	X		X	X			
PXP04NMF		16	X	X		X		X	
PXP08SPL	08	01	X			X	X	X	X
PXP08REQ		02	X			X	X	X	X
PXP08JNM		05	X			X			
PXP08UID		09	X			X			
PXP08BTS		1A	X		X			X	
PXP08IAB		1C	X	X		X			
PXP08IBT		24	X		X	X	X	X	X
PXP08ROS		25	X	X		X		X	X
PXP08BOS		27	X			X			
PXP08FB1		2B	X			X			
PXP08JNO		31	X			X			
PXP08XUA		4C					X		
PXP0CINS	0C	01	X	X	X	X	X	X	X
PXP0CIXF		02	X	X	X	X	X	X	X
PXP10PSP	10	05	X	X	X	X	X	X	X
PXP10SIE		06	X	X	X	X	X	X	X
PXP10CAA		03	X						
PXP10MST		07	X			X	X		

GCM Programming Example

Control Statements for Punching the Example

This example will punch the GCMEXAMP.Z into the VSE/POWER punch queue for further access.

```
* $$ JOB JNM=GCMJOB,CLASS=A,DISP=D
// JOB GCMJOB
// EXEC LIBR
   ACCESS S=IJSYSRS.SYSLIB
   PUNCH GCMEXAMP.Z
/*
/&
* $$ EOJ
```

GCM Programming Example Source Code

```
***********************************************************************
*        THIS EXAMPLE ILLUSTRATES THE RETRIEVAL OF JOB EVENT          *
*        MESSAGES. ALTHOUGH THE CODE THEREIN IS EXCLUSIVELY DESTINED  *
*        FOR ILLUSTRATION ONLY, IT CAN EASILY BE MADE EXECUTABLE      *
*        TO MEET THE USER'S NEEDS. SOME LOCATIONS REQUIRE THE SAME    *
*        CODING AS ALREADY USED IN THE 'PWRSASEX' EXAMPLE, SO THIS    *
*        CODE IS OMITTED AND ONLY A REFERENCE TO 'PWRSASEX' IS        *
*        INDICATED, WHERE THE RELEVANT CODING CAN BE FOUND UNDER      *
*        THE SAME LABEL.                                              *
*                                                                     *
*        ASSUME, THAT ANOTHER APPLICATION PROGRAM ALREADY SUBMITTED    *
*        JOBS TO VSE/POWER VIA THE SPOOL-ACCESS SUPPORT INTERFACE     *
*        USING THE 'QUEUE-EVENT-MESSAGE' OPTION, AND THESE            *
*        JOBS HAVE ALREADY FINISHED THEIR EXECUTION.                  *
*        THE APPLICATION PROGRAM SPECIFIED THE XPCC APPLID 'GCMAPPL'  *
*        AND THE SPOOL-ACCESS SUPPORT USERID 'THOMRAPP' TO SUBMIT     *
*        THE JOBS. THESE ID'S ARE NOW USED AGAIN TO RETRIEVE THE      *
*        RESULTING JOB EVENT MESSAGES. IN ORDER TO RETRIEVE           *
*        ALL JOB EVENT MESSAGES, THE FIELDS IN THE SPL FOR JOB        *
*        NAME (SPLGJB) AND JOB NUMBER (SPLGJN) ARE FILLED WITH        *
*        EIGHT BLANK CHARACTERS AND HEX ZERO, RESPECTIVELY.           *
*                                                                     *
*                                                                     *
*        THE EXAMPLE WILL SHOW:                                       *
*                                                                     *
*        1. HOW TO ESTABLISH A COMMUNICATION PATH TO VSE/POWER        *
*        2. HOW TO ISSUE A GCM-OPEN-KEEP REQUEST, WHICH COPIES        *
*           JOB EVENT MESSAGES FROM THE MESSAGE QUEUE TO THE          *
*           USER'S REPLY BUFFER.                                      *
*        3. HOW TO ISSUE A GCM-MORE SUBREQUEST IN ORDER TO RETRIEVE   *
*           STILL OUTSTANDING MESSAGES WHICH COULD NOT BE RETRIEVED   *
*           BECAUSE THE USER'S REPLY BUFFER WAS TOO SMALL TO HOLD ALL *
*           ELIGIBLE MESSAGES.                                        *
*        4. HOW TO ISSUE A GCM-REMOVE SUBREQUEST IN ORDER TO REMOVE   *
*           ALL ALREADY RETRIEVED MESSAGES FROM THE MESSAGE QUEUE     *
*        5. HOW TO TERMINATE THE COMMUNICATION TO VSE/POWER           *
*                                                                     *
***********************************************************************
GCMSAMP  CSECT                      START OF THIS SAMPLE PROGRAM
         BALR  R8,0                 GET START ADDRESS
         USING *,R8,R9              ESTABLISH ADDRESSABILITY
         SPACE 2
         LA    R9,4095(,R8)         LOAD SECOND BASE REGISTER WITH
         LA    R9,1(,R9)            CONTENTS OF FIRST + 4096
         SPACE 2
         LA    R4,OWNXPCCB          GET ADDR OF CROSS PART. CONTROL BLK
         USING IJBXPCCB,R4          ESTABLISH ADDRESSABILITY FOR DSECT
         SPACE 2
         LA    R5,IJBXSUSR          GET ADDR OF USER DATA TO BE SENT
         USING PXUUSER,R5           ESTABLISH ADDRESSABILITY FOR DSECT
         SPACE 2
         LA    R6,IJBXRUSR          GET ADDR OF RECEIVED USER DATA
         USING PXPUSER,R6           ESTABLISH ADDRESSABILITY FOR DSECT
         SPACE 2
         LA    R7,OWNSPL            GET ADDR OF SPL
         USING OWNSPLDS,R7          ESTABLISH ADDRESSABILITY FOR DSECT
         EJECT
***********************************************************************
**          >> IDENTIFY GCMSAMP VSE/AF XPCC USER <<                  **
***********************************************************************
         SPACE 1
IDENT    DS    0H
         SPACE 1
         XPCC  XPCCB=(R4),FUNC=IDENT   IDENTIFY 'GCMAPPL' TO AF-XPCC
         SPACE 1
```

```
*        FOR ERROR CHECKING, SEE PWRSASEX (IDENT) -------------------->
***********************************************************************
**      >> ESTABLISH THE XPCC CONNECTION TO VSE/POWER <<             **
***********************************************************************
         SPACE 1
CONCT    DS    0H
         SPACE 1
         XPCC  XPCCB=(R4),FUNC=CONNECT        CONNECT TO VSE/POWER
         SPACE 1
*        FOR ERROR CHECKING, SEE PWRSASEX (CONCT) -------------------->
         SPACE 1
         EJECT
***********************************************************************
**      >> RETRIEVE JOB EVENT MESSAGES BY MEANS OF GCM-OPEN-KEEP <<  **
***********************************************************************
         SPACE 1
GCMA1    DS    0H
         PWRSPL TYPE=UPD,SPL=OWNSPL,REQ=GCM
         SPACE 2
         MVI   PXUBTYP,PXUBTSPL    INDICATE BUFFER TYPE = SPL
         MVI   PXUACT1,0           CLEAR ALL OTHER BYTES IN PXUUSER,
         MVI   PXUSIGNL,0          WHICH MAY BE CHANGED BY THE USER
         STCM  R7,M7,IJBXADR       INSERT SPL ADDRESS AS BUFFER ADDR.
         LA    R3,SPLGLEN          LOAD LENGTH OF SPL
         ST    R3,IJBXBLN          INSERT BUFFER LENGTH INTO XPCCB
         SPACE 1
*        SPECIFICATION OF THE GCM REQUEST TYPE AND SELECTION CRITERIA
         SPACE 1
         MVI   SPLGFB1,SPLGF1KM    SPEC. GCM-OPEN-KEEP
         MVC   SPLGJB,JOBNBLNK     SPEC. BLANK JOB NAME: ANY NAME
         MVC   SPLGJN,JOBNUMB      SPEC. JOB NUMBER: ANY NUMBER
         SPACE 1
*        ISSUE THE GCM-OPEN-KEEP REQUEST AND RETRIEVE THE MESSAGES.
*        IF THERE ARE MORE MESSAGES TO RETRIEVE, ISSUE THE GCM-MORE
*        SUBREQUEST AS LONG AS THERE ARE MORE MESSAGES AVAILABLE.
         SPACE 1
GCMMORE  DS    0H                  DO UNTIL EOD OR FAILURE
         BAL   RD,SENDR              ISSUE THE REQUEST
         CLI   PXPRETCD,PXPRCOK      WAS VSE/POWER RET. CODE ZERO?
         BNE   REQFAIL               ..NO, GO TO HANDLE FAILURE
         CLI   PXPFBKCD,PXP00OK      WAS VSE/POWER FDBK CODE ZERO?
         BE    GCMA2                 ..YES, MORE TO RETRIEVE
         CLI   PXPFBKCD,PXP00EOD     FB WAS EOD?
         BE    GCMA2                 ..YES, PROCESS BUFFER
         B     REQFAIL               ..NO, GO TO HANDLE FAILURE
         SPACE 1
*        PROCESS RETURNED MESSAGE BUFFER
         SPACE 1
GCMA2    DS    0H                    PROCESS MESSAGE BUFFER
         BAL   RE,BUFPROC            PROCESS RETURNED MSG BUFFER
         CLI   PXPFBKCD,PXP00EOD     FB WAS EOD?
         BE    GCMREM                ..YES, GO TO REMOVE THE MSGS
         SPACE 1
*        SET UP THE GCM-MORE SUBREQUEST
         SPACE 1
         MVI   PXUACT1,PXUATGCM      SIGNAL GCM-MORE
         MVI   PXUBTYP,0             SIGNAL NULL BUFFER
         XC    IJBXBLN,IJBXBLN       SET UP NULL BUFFER
         B     GCMMORE             END UNTIL EOD OR FAILURE
         SPACE 2
*        ISSUE THE GCM-REMOVE SUBREQUEST
         SPACE 1
GCMREM   DS    0H                  REMOVE THE MSGS
         MVI   PXUBTYP,0           SIGNAL NULL BUFFER
         MVI   PXUACT1,PXUATDEL    SIGNAL GCM-REMOVE
         XC    IJBXBLN,IJBXBLN     SET UP NULL BUFFER
         BAL   RD,SENDR            ISSUE THE REQUEST
```

```
         CLI   PXPRETCD,PXPRCOK    WAS VSE/POWER RET. CODE ZERO?
         BNE   REQFAIL             ..NO, GO TO HANDLE FAILURE
         CLI   PXPFBKCD,PXP00EOD   ALL MSG'S DELETED?
         BE    DISCT               ..YES, GO TO DISCONNECT
         B     REQFAIL             ..NO, GO TO REPORT ERROR
         SPACE 2
         EJECT
***********************************************************************
*        >> PROCESS JOB EVENT MESSAGES CONTAINED IN YOUR BUFFER
***********************************************************************
BUFPROC  DS    0H                  GET BEGIN OF REPLY BUFFER
         SR    R0,R0               SET R0 TO ZERO
         CLM   R0,M7,IJBXSLN       NO MORE DATA TO DISPLAY?
         BE    BUFPROX             ..YES, GO TO ROUTINE EXIT
         LA    BUFPTR,REPLBUF      POINT TO REPLY BUFFER
         SR    BUFLN,BUFLN         INIT COUNT FOR REM.UNPROC. BUFFER
         ICM   BUFLN,M7,IJBXSLN    GET LENGTH OF DATA TO BE PROCESSED
         SPACE 1
BUFPR0   DS    0H                  SET BUFFER POINTER TO 1ST MSG
         USING RECPRFIX,BUFPTR     GET DSECT OF RECORD LAYOUT
         LH    R2,RECLNGTH         GET LENGTH OF FIRST/NEXT DATA REC.
         LR    RF,R2               SAVE RECLENGTH
         CLI   RECTYPE,RECTFJCM    JOB COMPLETION MSG?
         BNE   BUFPRJG             ..NO, MUST BE JOB GENERATION MSG
         SPACE 1
*        PROCESS FIXED FORMAT JOB COMPLETION MESSAGE
         SPACE 1
BUFPRJC  DS    0H                  SET BUFFER POINTER TO 1ST MSG
         LA    BUFPTR,RECPRFXL(,BUFPTR)     SKIP RECORD PREFIX
         LR    R3,BUFPTR           GET MESSAGE ADDRESS
         USING JCMDS,R3            MAKE F.F. MSG ADDRESSABLE
         SPACE 1
* -------------------------------------------------------------------
*   >>   INCLUDE HERE YOUR CODING TO PROCESS THE DATA OF ONE JOB
*   >>   COMPLETION MESSAGE (F.F. JCM)
* -------------------------------------------------------------------
         DROP  R3                  DROP F.F. JCM ADDR'Y
         B     BUFPR1              GO TO MOVE BUFFER POINTER
         SPACE 2
*        PROCESS FIXED FORMAT JOB GENERATION MESSAGE
         SPACE 1
BUFPRJG  DS    0H                  SET BUFFER POINTER TO 1ST MSG
         LA    BUFPTR,RECPRFXL(,BUFPTR)     SKIP RECORD PREFIX
         LR    R3,BUFPTR           GET MESSAGE ADDRESS
         USING JGMDS,R3            MAKE F.F. MSG ADDRESSABLE
         SPACE 1
* --------------------------------------------------------------------
*   >>   INCLUDE HERE YOUR CODING TO PROCESS THE DATA OF ONE JOB
*   >>   GENERATION MESSAGE (F.F. JGM)
* --------------------------------------------------------------------
         DROP  R3                  DROP F.F. JGM ADDR'Y
         SPACE 2
BUFPR1   DS    0H                  MOVE BUFFER POINTER
         LR    R2,RF               RESTORE REC LENGTH
         LA    R1,RECPRFXL(,R2)    CALC. LENGTH OF RECORD INCL. PREFIX
         SR    BUFLN,R1            CALC. LENGTH OF DATA STILL TO PROC.
         LA    BUFPTR,0(R2,BUFPTR) POINT TO NEXT RECORD
         LTR   BUFLN,BUFLN         ALL DATA IN BUFFER PROCESSED?
         BNZ   BUFPR0              ..NO, GO TO PROCESS NEXT DATA REC.
         SPACE 1
BUFPROX  DS    0H                  ROUTINE EXIT
         BR    RE                  RETURN TO CALLER
         EJECT
***********************************************************************
**    >> ROUTINE TO HANDLE REQUEST FAILURES  <<                      **
***********************************************************************
REQFAIL  DS    0H
```

```
*        ESTABLISH CODING TO HANDLE ANY REQUEST FAILURES. TERMINATE
*        OR CONTINUE THE REQUEST, WHATEVER IS REQUIRED.
***********************************************************************
**    >> DISCONNECT THE XPCC COMMUNICATION LINK TO VSE/POWER <<      **
***********************************************************************
         SPACE 1
DISCT    DS    0H
         XPCC  XPCCB=(R4),FUNC=DISCONN   DISCONNECT LINK TO VSE/POWER
         SPACE 1
         LTR   RF,RF               WAS DISCONNECT SUCCESSFUL, RF='00'?
         BZ    TERMN               ..YES CONTINUE WITH XPCC TERMINATION
*        FOR ERROR PROCESSING, SEE PWRSASEX (DISCT) ------------------>
***********************************************************************
**    >> TERMINATE INTERACTION WITH THE VSE/AF XPCC SUPPORT <<       **
***********************************************************************
TERMN    DS    0H
         XPCC XPCCB=(R4),FUNC=TERMIN   TERMINATE CROSS PART. INTERFACE
         LTR   RF,RF               DID WE GET A ZERO RET-CODE ?
         BZ    FINEND              ..YES, GO TO NORMAL EOJ MACRO
*        FOR ERROR PROCESSING, SEE PWRSASEX (TERMN) ------------------>
***********************************************************************
**                 >> TERMINATE MESSAGE RETRIEVAL <<                 **
***********************************************************************
         SPACE 1
FINEND   DS    0H                  NORMAL TERMINATION
         EOJ                       NORMAL END OF GCMSAMP PROGRAM
***********************************************************************
**          >> CENTRAL XPCC SENDR ROUTINE <<                         **
***********************************************************************
SENDR    DS    0H
*        FOR CODING OF A SENDR REQUEST, SEE PWRSASEX (SENDR) --------->
         SPACE 2
***********************************************************************
**               D E F I N I T I O N S                               **
***********************************************************************
         SPACE 2
***********************************************************************
*        STORAGE RESERVATION FOR  XPCC SEND AND REPLY BUFFER         *
***********************************************************************
         SPACE 1
SENDBUF  DS    CL400           BUFFER USED FOR XPCC SENDR TO VSE/POWER
REPLBUF  DS    CL500           BUFFER FOR RECEIPT OF DATA FROM VSE/POWER
         SPACE 2
***********************************************************************
*        >> CROSS PARTITION CONTROL BLOCK <<                         *
***********************************************************************
         SPACE 1
OWNXPCCB XPCCB APPL=GCMAPPL,TOAPPL=SYSPWR,                             *
               BUFFER=(SENDBUF,400),REPAREA=(REPLBUF,500)
         SPACE 2
**********************************************************************
**       >>      GENERATE   S P L        <<                         *
**********************************************************************
         SPACE 1
OWNSPL   PWRSPL TYPE=GEN,USERID=THOMRAPP,PRFX=OWN
         EJECT
***********************************************************************
*        DUMMY SECTION OF  VSE/POWER SPOOL PARAMETER LIST (SPL)      *
***********************************************************************
         SPACE 1
OWNSPLDS PWRSPL TYPE=MAP
         EJECT
***********************************************************************
*        DUMMY SECTION OF  CROSS PARTITION CONTROL BLOCK  (XPCCB)    *
***********************************************************************
         SPACE 1
         MAPXPCCB
```

```
         EJECT
***********************************************************************@
*        >> SPECIFICATION OF SELECTION CRITERIA <<                     @
***********************************************************************@
JOBNBLNK DC    CL8'        '     DATA FOR SELECTION CRITERIA          @
JOBNUMB  DC    AL2(0)            DATA FOR SELECTION CRITERIA          @
***********************************************************************
*        EQUATES                                                      *
***********************************************************************
         SPACE 1
M7       EQU   7                   MASK BIT SETTING
BUFPTR   EQU   10                  USE RA AS BUFPOINTER
BUFLN    EQU   12                  USE RC TO CALC REMAINING BUFLEN
R0       EQU   0                   WORK REGISTER
R1       EQU   1                   WORK REGISTER + USED BY PWRSPL MACRO
R2       EQU   2                   WORK REGISTER
R3       EQU   3                   WORK REGISTER
R4       EQU   4                   ADDR REG FOR XPCCB DSECT
0
R6       EQU   6                   ADDR REG FOR RECEIVED USER DATA
R7       EQU   7                   ADDR REG FOR SPL DSECT
R8       EQU   8                   FIRST BASE REGISTER OF GCMSAMP
R9       EQU   9                   SECOND BASE REGISTER OF GCMSAMP
RA       EQU   10                  WORK REGISTER
RB       EQU   11                  WORK REGISTER
RC       EQU   12                  WORK REGISTER
RD       EQU   13                  BRANCH AND LINK REGISTER FOR SENDR
RE       EQU   14                  BRANCH AND LINK REG. FOR BUFPROC
RF       EQU   15                  MACRO CALL RETURN CODE REGISTER
         SPACE 1
         END
```

Chapter 11. Supporting I/O Devices Via Device Driving Systems

The external device support is a special application of the spool-access support described in Chapter 6, “Introduction to Spool-Access Support,” on page 57 and the following chapters. Therefore, this description is based on the preceding chapters.

The support shifts the control for writing spooled output to a device from VSE/POWER to a device-driving system (DDS), for example, a CICS spooler or PSF. This device-driving system may run in a partition under or outside the control of VSE/POWER. The support allows you, for example, to process output spooled to the LST or PUN queue on a device which is not supported by VSE/POWER.

Using the support requires you to implement extensive coding of your own in your program. This coding must be done in assembler language.

The coding required in your program is illustrated in

- this publication by Table 50 on page 176 to Table 58 on page 194, showing your program steps together with a "comment" column, which explains "how to code" and "what VSE/POWER does".
- the *VSE/POWER Diagnosis Reference Manual* by figures that use the "communication protocol" description for your program steps and how VSE/POWER reacts in each case. You may find it helpful to consult this publication in addition.

This chapter briefly discusses the operational concepts of the support and describes how to use it. The macros you need to implement the support in your program are documented in Chapter 12, “Spool-Access Support Macros,” on page 211.

For more information on the return and feedback codes see Chapter 14, “Return and Feedback Codes and Their Meanings,” on page 297.

To make the description even more understandable and to facilitate entry into your own code, you can request the "DDSIM" programming example from the VSE/POWER development libraries by sending an e-mail to `L2POWER@de.ibm.com`.

Concepts

Programming Prerequisites

Figure 9 on page 172 shows how a device driving system communicates with VSE/POWER. Before a subsystem-controlled device can be started for output of spooled data, this subsystem must:

1. Identify itself to the system.
2. Issue one or more connect-any requests, one per device that is to be used for the processing of spooled output. A connect-any request ensures that VSE/POWER can establish a communication path when a PSTART command for the device is issued.

User Responsibilities

The subsystem must provide for all of the services normally available for a device under VSE/POWER control. This includes services such as device recovery, measurement techniques for performance and accounting, and protection of spooled data after VSE/POWER has passed this data to the subsystem.

Operational Overview

Following is an overview of the operational steps involved in writing spooled output to a device under subsystem control. This overview assumes that the device-owning subsystem is up and running. It further assumes that the output device to be used is ready.

1. VSE/POWER processes a PSTART command for the device, for example:
   ```
   PSTART DEV,PLOT1,GRAPHAPP,G,...
   ```
   The command causes VSE/POWER to activate a device service task which establishes a communication path to the subsystem named GRAPHAPP.

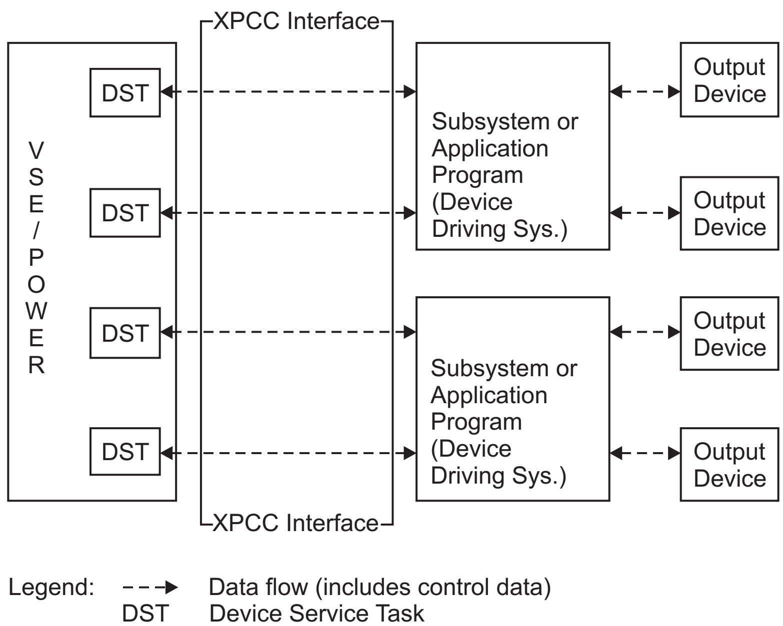


Figure 9. External Device Support Overview

2. When the communication path is established, the device owning subsystem passes to VSE/POWER a request for a device order.
3. In response to the request, VSE/POWER passes to the subsystem a “start device” order. This order includes all of the control values that were specified in the above PSTART command.
4. The subsystem, after having confirmed the order by passing an order-response record, would normally issue a GET-GENERIC request. An example of such a request is given below:
   ```
   PWRSPL TYPE=UPD,CLASS=G,MODE=GENERIC,QUEUE=LST,REQ=GET,SPL=MYSPL...
   ```

   Passing this updated SPL to VSE/POWER via XPCC FUNC=SENDR causes VSE/POWER to return to your program any output queued with the specified class (G in the above example) of the LST queue for the device PLOT1.

The retrieval of a complete queue entry requires the subsystem to issue a series of Get spool data requests with XPCC FUNC=SENDR. Per request, VSE/POWER passes a unit of transfer, one or more records of data, to your program's reply buffer.

Note that before your program can set up an XPCC FUNC=SENDR request, it must always clear the XPCCB User Data IJBXSUSR.

5. The subsystem writes every unit of transfer to the output device selected by the PSTART command.
6. When the processing of a queue entry is complete, the subsystem issues a close request followed by another GET-GENERIC request to open the retrieval of the next eligible queue entry.

The above sequence of operational steps continues as long as there is work to do. This sequence, although not all inclusive, shows that your program must synchronize its operation with VSE/POWER primarily by:

1. Picking up and analyzing any device order that VSE/POWER may pass.
2. Responding to a device order by passing to VSE/POWER the corresponding order-response record. This response record must indicate how your program is going to handle the device order.

Shared Spooling Considerations

For operation with external device support in a shared spooling environment, the following restrictions exist:

- Only one system operator can control your program's output devices: the operator of the system on which your program is running.
- Messages passed to VSE/POWER for routing to a user of one of the other sharing systems cannot be forwarded to this user by VSE/POWER.

The remaining sections of this chapter discuss the sequences of the coding required to ensure proper handling of spooled output. These sequences are discussed as part of the applicable communication and device-control functions.

IBM recommends that you obtain a listing of the DSECTS that are generated by the assembly of the PWRSPL TYPE=MAP macro and that you have this listing readily available at your finger tips. This may be helpful for the study of the chapter.

Setting Up a Communication Path

Your program must initiate the setup of required communication paths. To do this, provide code in your program to:

1. Identify your program to the system.

   You do this by way of an XPCC macro specifying FUNC=IDENT.
2. Initiate setting up a communication path, one per device.

   You do this by way of an XPCC FUNC=CONNECT with TOAPPL=ANY specified in the related XPCCB macro.

   In your program, you can issue as many XPCC FUNC=CONNECT requests as you have devices to control for the processing of spooled output. A connect request must be complete before you can issue the next one.

For more information about establishing a communication path, see “Setting Up a Communication Path” on page 59. “Setting Up Several Communication Paths” on page 64 describes how to establish several communication paths.

Starting a Device

Starting a device is triggered by a PSTART command issued by one of the following:

The central operator

An authorized subsystem administrator

Via PNET

In processing the command, VSE/POWER tries to set up a communication path within two minutes. If VSE/POWER cannot set up the path within this time, then the originator of the PSTART command gets a message.

VSE/POWER expects this originator to react to the message as follows:
- if the device driving subsystem cannot establish a communication path to VSE/POWER for device activation, issue the PSTOP device command, or
- wait until the subsystem is prepared for the device setup.

Your program must include code which does the following (assuming that you have properly initiated the setup of a communication path):
1. Waits for the communication path to be set up.

   You do this by checking whether the system has posted the connect ECB (field IJBXCECB of the applicable XPCCB).
2. Passes to VSE/POWER a request for a device order.

   You do this by issuing an XPCC FUNC=SENDR request which:
   - Passes a null buffer (IJBXBLN set to zero).
   - Has XPCCB bytes set as follows:

     PXUACT1 to PXUATROR

     PXUBTYP to zero

   and by checking for successful completion or, if necessary, by analyzing return information that the system may have set in the fields IJBXRETC and later in IJBXREAS of the XPCCB.
3. Analyzes the start device order which VSE/POWER passes in the reply buffer for the communication path.
4. Passes to VSE/POWER the corresponding order-response record.

   To do this, issue an XPCC FUNC=SENDR request with this record set up in the communication path's send buffer. Before you issue this request, clear the XPCCB User Data IJBXSUSR.

For a more detailed discussion of the start-device sequence, see the related sections that follow.

Processing a Start-Device Order

If Device Can Be Started

Refer to Table 50 on page 176, the coding sequence for starting a device under subsystem control. For the layout and contents of order-control and response records, see the section “Processing of Order-Control Records and Signals” on page 192.

If Device Cannot Be Started

Your program may not be prepared to process output on the device as requested. You must indicate this and give a reason by setting a return-and-feedback code in your order-response record for one of the following, for example:

Device unknown

Device in use (busy)

Device out of service

A return code other than X'00' causes VSE/POWER to break the connection. For details about these codes, see the section “Start-Device Order” on page 199.

Based on your program's control data, VSE/POWER builds a message and routes it to the command originator.

Starting a Device with 'Set Logical Destinations'

If your program does not use a set-logical-destinations order (see “Set-Logical-Destination Order” on page 205), VSE/POWER takes the specified device name (PLOT1 for example) as the only valid destination name for the device.

If you use a set-logical-destinations order, your program can define to VSE/POWER up to eight logical destination names for one device. Assume that a device has been started in your program with a device name of PLOT1. You could then request VSE/POWER by a set-logical-destinations order to route, via the path for PLOT1, output to the following destinations, for example:

D121OUT

D122OUT

D123OUT

and so on

If any of these logical destinations is specified as destination user of an output, then VSE/POWER routes this output to the external device named PLOT1.

However, if the device name used in the PSTART command is to be used as user ID for routing output further on, that name must be included in the list of logical destinations.

Note: The logical destination name LOCAL returns queue entries either

1. destined for local processing or destined for the user ID LOCAL.
2. do not use R000 thru R250 as logical destination, since these are reserved for RJE userid's.

Table 50. Code for Starting an External Device Sequence

Coding in your application program	Comments
... ... XPCC FUNC=CONNECT Check the return codes in register 15 and in the XPCCB (byte IJBXRETC). WAIT IJBXCECB	 Connect with TOAPPL=ANY The communication path exists when the ECB is posted. Field IJBXTOAP of XPCCB contains 'SYSPWRD'.
Request a device order to be passed XPCC FUNC=SENDR Check the return codes as shown above. WAIT IJBXSECB Check the VSE reason codes in XPCCB byte IJBXREAS. Check the VSE/POWER return and feedback codes in XPCCB bytes PXPRETCD and PXPFBKCD, respectively. Analyze the device order	A device order is in your program's reply buffer when the ECB is posted. Normally, your program finds a start device order after successful setup of a communication path.
Respond to the order XPCC FUNC=SENDR Check the return codes as shown above. WAIT IJBXSECB Check the VSE reason codes in XPCCB byte IJBXREAS. Check the VSE/POWER return and feedback codes as shown above.	VSE/POWER has finished processing your order-response record and returned a null buffer when the ECB is posted. If these codes indicate successful processing of the order-response record, then VSE/POWER is ready to process GET-service requests.

The coding sequence for a device start with setting logical destinations is the same as for a normal device start (see Table 50). In addition, however, your program must pass to VSE/POWER a set-logical-destinations order. You do this after VSE/POWER has successfully processed your order-response record for the start-device order. VSE/POWER responds to your order by passing an order-response record to your program's reply buffer.

For the layout and contents of control records, see the section "Processing of Order-Control Records and Signals" on page 192.

Processing Spooled Output

When your program is ready to process an output queue entry, it should issue a generic GET-OPEN request. To do this, pass to VSE/POWER an SPL for which you defined, for example, the following:

```
                                                       Column 72----
                                                                   |
PWRSPL TYPE=UPD,SPL=(4),CLASS=G,MODE=GENERIC,                      C
       QUEUE=LST,REQ=GET
```

Then VSE/POWER retrieves from the accessed queue (LST in the example) the first queue entry that it finds to have:

- Class G assigned
- A disposition of D or K
- A user ID matching one of the logical destinations of the device

In response to your open-service request, VSE/POWER passes to your program's reply buffer an SPL which describes the queue entry's characteristics. Your program must analyze this SPL and decide whether VSE/POWER is to proceed with data retrieval or whether any other action is to be initiated.

The subsystem, your program, has to handle certain situations which VSE/POWER handles when processing the output of spooled data on a local device. The handling of these situations is normally triggered by a device order passed to your program by VSE/POWER. Of course, the handling of a device failure, should one occur, cannot be triggered by VSE/POWER. Some of these situations are discussed in sections as indicated below; they should give you a feel for the involved programming effort:

- No selectable entry in the accessed queue – See “Handling a No-Selectable-Entry Situation.”
- A device setup is required to process the output – See “Handling a Device-Setup Situation” on page 178.
- Output processing is to be canceled – See “Canceling Output Processing” on page 183.
- VSE/POWER-queued device orders or signals are to be requested – See “Requesting an Order or a Signal” on page 183.

Note: No password checking is done for a queue entry that is to be processed by a subsystem for output under subsystem control.

Handling a No-Selectable-Entry Situation

If there is no selectable queue entry, VSE/POWER informs the system operator about this. In addition, it informs your program by way of return-and-feedback codes in the VSE/POWER-set user area of the XPCCB. VSE/POWER then waits for one of the following:

An order from your program (message or set-logical-destination).

A ‘wait-for-order/signal’ request from your program.

A command from the operator.

A selectable output queue entry to be queued.

Table 51 on page 178 shows the sequence of the coding which you should provide in your program to cover the situation. Instead of passing a wait-for-order/signal indication to VSE/POWER, your program may take either of the actions below.

- Give up the communication path (by an XPCC FUNC=DISCONN).
- Define or change one or more of the logical destination names for the device (by a set-logical-destination order), followed by another generic GET request.

Handling a Device-Setup Situation

Your program should analyze the verification SPL which VSE/POWER passes after the Get-service open request. As a result of this analysis, your program may have to initiate a device setup. The operational steps for this setup normally are as follows:

1. Your program passes a send-message order control record.

   This order instructs VSE/POWER to route the included message to the destination given in the order. VSE/POWER forwards the message to this destination, normally the operator responsible for the output device which is to be set up.

   Your order-control record may request VSE/POWER to hold a copy of the message in storage: the message may fail to reach its destination, and VSE/POWER's device-service task may therefore be operator bound. A copy of the message is helpful in this case; it enables the central operator to redisplay the message by means of a PDISPLAY M command. For more information about processing a send-message order, see “Send-Message Order” on page 205.
2. Your program waits for the reactivation of this output processing.

   The program does this by passing to VSE/POWER a wait-for-order/signal request (XPCCB bytes set as follows: PXUACT1 to PXUATWFR; PXUBTYP to zero).

   When a device order or a signal gets queued for the communication path to your program, then VSE/POWER passes this order or signal.

Table 51. Code for a “No Entry Available” Situation Sequence

Coding in your application program	Comments
... ... Open GET service XPCC FUNC=SENDR Check the return codes in register 15 and in the XPCCB (byte IJBXRETC).	This should be a generic GET-OPEN request. VSE/POWER expects an (updated) SPL in your program's send.
WAIT IJBXSECB Check the VSE reason codes in the XPCCB byte IJBXREAS. Check the VSE/POWER return and feedback codes in XPCCB bytes PXPRETCD and PXPFBKCD, respectively.	A return SPL is in your program's reply buffer when the ECB is posted, provided an eligible queue entry was found. The feedback code (byte PXPFBKCD) is set to PXP04NOF if VSE/POWER cannot find a selectable queue entry. The remainder of the sequence chart applies to this case.

Table 51. Code for a "No Entry Available" Situation Sequence (continued)

Coding in your application program	Comments
Pass a wait-for-order/signal request XPCC FUNC=SENDR Check the return codes as shown above. WAIT IJBXSECB Check the VSE reason codes as shown above. Check the VSE/POWER return and feedback codes as shown above. Analyze the order/signal.	Pass a null buffer to VSE/POWER and be sure the wait-for-order/signal flag (PXUACT1 set to PXUATWFR) is set in the XPCCB (see Note). VSE/POWER has passed an order or a signal when the ECB is posted. VSE/POWER passes an output-arrived signal as soon as a selectable queue entry is queued. Let's assume that VSE/POWER did pass the signal. This means that VSE/POWER is ready to accept a GET-service request via the communication path.
Open GET service XPCC FUNC=SENDR	A retry of the originally passed generic GET-OPEN request. Again, a selectable queue entry may not be available for processing. By the time VSE/POWER processes your program's request, the queued entry may have been manipulated from another source. **Note**: Since no selectable queue entry is available and no 'order pending' is indicated by VSE/POWER, your program should use the PXUATWFR request. This results in a VSE/POWER wait for the next order or signal, while a PXUATROR request would return immediate information about the availability of an order/signal.

3. Output processing is reactivated.

   The operator issues a PGO command to indicate that the required setup work is done. This makes VSE/POWER queue a reactivation-device order so that it can be passed to your program.

   Your program cannot reactivate output processing until VSE/POWER has passed a reactivate-device order. If VSE/POWER passes a device order other than reactivate (or setup), your program must respond to this order and reissue the wait-for-order/signal request.

Table 52 on page 180 shows the sequence of the coding which you should provide to cover the needs of a device setup and a reactivation of output processing. For more details about the processing of device orders, order-response records, and device signals, see the section "Processing of Order-Control Records and Signals" on page 192.

Table 52. Code for Device Setup and Reactivation Sequence

Coding in your application program	Comments
... ... Open GET service XPCC FUNC=SENDR Check the return codes in register 15 and in the XPCCB (byte IJBXRETC). WAIT IJBXSECB Check the VSE reason codes in the XPCCB byte IJBXREAS. Check the VSE/POWER return and feedback codes in XPCCB bytes PXPRETCD and PXPFBKCD, respectively. Analyze the verification SPL.	This should be a generic GET-OPEN request. VSE/POWER expects an (updated) SPL in your program's send buffer. A verification SPL is in your program's reply buffer when the ECB is posted. The remainder of the chart assumes that the indicated device characteristics require a device setup.
Pass a send-message order XPCC FUNC=SENDR Check the return codes as shown above. WAIT IJBXSECB Check the VSE reason codes as shown above. Check the VSE/POWER return and feedback codes as shown above.	The order tells VSE/POWER where to route the message which is part of the order-control record. VSE/POWER's order response record is in your program's reply buffer when the ECB is posted.
Pass a wait-for-order/signal request XPCC FUNC=SENDR Check the return codes as shown above.	Pass a null buffer to VSE/POWER and be sure the "wait for order/signal" flag is set in the XPCCB.
WAIT IJBXSECB Check the VSE reason codes as shown above. Check the VSE/POWER return and feedback codes as shown above. Analyze the device order.	VSE/POWER has passed an order or a signal when the ECB is posted. Let's assume that VSE/POWER passed a setup-device order. It indicates the number of pages the operator asks your program to retrieve from VSE/POWER and pass to the device for setup purposes.

Table 52. Code for Device Setup and Reactivation Sequence (continued)

Coding in your application program	Comments
Pass an order-response record XPCC FUNC=SENDR Check the return codes in register 15 and in the XPCCB (byte IJBXRETC). WAIT IJBXSECB Check the VSE reason codes in the XPCCB byte IJBXREAS. Check the VSE/POWER return and feedback codes in XPCCB bytes PXPRETCD and PXPFBKCD, respectively.	VSE/POWER has processed the response record and returned a null buffer when the ECB is posted. Required programmed action: 1. Request VSE/POWER to pass the defined number of setup pages. 2. Reactivate normal processing when the setup action is complete.
GET spool data request XPCC FUNC=SENDR Check the return codes as shown above. WAIT IJBXSECB Check the VSE reason codes as shown above. Check the VSE/POWER return and feedback codes as shown above. Process the records in your reply buffer. If end of last setup page, proceed to the next step; else **return to the beginning of this step.**	When the ECB is posted, your program's reply buffer is filled with spooled output records retrieved from the accessed queue entry. The data being passed may have to be replaced by strings of Xs.
Pass the setup-processed signal XPCC FUNC=SENDR Check the return codes as shown above. WAIT IJBXSECB Check the VSE reason codes as shown above. Check the VSE/POWER return and feedback codes as shown above.	Your program passes a null buffer with PXUSIGNL of the XPCCB set to PXUSSET. This causes VSE/POWER to reset its retrieval pointers to the beginning of the queue entry being processed. VSE/POWER has processed the signal and returned a null buffer when the ECB is posted.

Table 52. Code for Device Setup and Reactivation Sequence (continued)

Coding in your application program	Comments
Pass a wait-for-order/signal request XPCC FUNC=SENDR Check the return codes in register 15 and in the XPCCB (byte IJBXRETC). WAIT IJBXSECB Check the VSE reason codes in the XPCCB byte IJBXREAS. Check the VSE/POWER return and feedback codes in XPCCB bytes PXPRETCD and PXPFBKCD, respectively. Analyze the order/signal	Your program passes a null buffer and the wait-for-order/signal flag in the XPCCB. VSE/POWER has passed an order or a signal when the ECB is posted. Let's assume that VSE/POWER passed a reactivate-device order.
Pass an order-response record XPCC FUNC=SENDR Check the return codes as shown above. WAIT IJBXSECB Check the VSE reason codes as shown above. Check the VSE/POWER return and feedback codes as shown above.	VSE/POWER is ready to accept GET service requests and returned a null buffer when the ECB is posted (if VSE/POWER's return and feedback codes are OK).
GET spool data request XPCC FUNC=SENDR Check the return codes as shown above. WAIT IJBXSECB Check the VSE reason codes as shown above. Check the VSE/POWER return and feedback codes as shown above. Process the data passed by VSE/POWER. If more data is to be processed, **return to the beginning of this step**. Else proceed.	When the ECB is posted, your program's reply buffer is filled with spooled output records retrieved from the accessed queue entry.
Pass a close request XPCC FUNC=SENDR Check the return codes in register 15 and in the XPCCB (byte IJBXRETC). WAIT IJBXSECB Check the VSE reason codes in the XPCCB byte IJBXREAS. Check the VSE/POWER return and feedback codes in XPCCB bytes PXPRETCD and PXPFBKCD, respectively.	Your program passes a null buffer with the XPCCB bytes set as follows: PXUACT1 to PXUATRQS PXUBTYP to zero When the ECB is posted, VSE/POWER has disposed of the just processed queue entry in accordance with the assigned disposition: D - The entry is deleted K - The entry's disposition is changed to L.

Table 52. Code for Device Setup and Reactivation Sequence (continued)

Coding in your application program	Comments
Process the next selectable queue entry or end the retrieval of output.	

Canceling Output Processing

Output processing is to be canceled when VSE/POWER receives a PFLUSH command for the device under your program's control. The command may request this cancelation with or without a HOLD specification.

For the PFLUSH command, VSE/POWER builds and queues a flush-device order. This order is passed to your program in response to a return-order/signal request.

Your program may delay the requested cancelation until a certain point in its processing; it may ignore the order by returning a not-accepted response. Normally, however, a subsystem would handle the device order as shown:

- In Table 53 on page 184 for a PFLUSH without a HOLD specification.
- In Table 54 on page 185 for a PFLUSH with a HOLD specification.

  If HOLD is specified, your program should continue output processing until a meaningful boundary (end of a page, for example) is reached. This may require your program to request a certain number of output records even after VSE/POWER passed the flush-device order. In addition, your program should request a checkpoint to be taken before it stops processing for the output that is to be canceled.

If a cancel message is to be written at the end of the canceled output, your program must build the message and write it to the device.

Requesting an Order or a Signal

VSE/POWER chains and passes device orders (or signals), using the first-in/first-out method. When it chains an order or signal, VSE/POWER indicates this by setting the user byte PXPINFO to PXPIORD. Your program should monitor the presence of a device order by testing this byte along with the VSE/POWER return-and-feedback codes.

For VSE/POWER to pass the order next in line, you must code the following in your program:

- If no order is queued and your program needs a certain order to continue –

  A wait-for-order/signal request. You do this by passing to VSE/POWER an XPCC FUNC=SENDR with a null buffer and PXUACT1 set to PXUATWFR. You would use this method, for example, in a device-setup situation after your program has passed a send-message order.
- If an order is queued –

  A return-order/signal request. You do this by passing to VSE/POWER an XPCC FUNC=SENDR with a null buffer and PXUACT1 set to PXUATROR.

Whenever VSE/POWER passes to you a device order, it expects you to return (in your send buffer) an order-response. For more information about the processing of orders, see the section "Processing of Order-Control Records and Signals" on page 192.

Table 53. Code for a PFLUSH without HOLD Sequence

Coding in your application program	Comments
... ...	Assumption: During GET data processing the PXPIORD (order signal queued) indication is set in the your program's XPCCB.
Pass a return-order/signal request XPCC FUNC=SENDR Check the return codes in register 15 and in the XPCCB (byte IJBXRETC). WAIT IJBXSECB Check the VSE reason codes in the XPCCB byte IJBXREAS. Check the VSE/POWER return and feedback codes in XPCCB bytes PXPRETCD and PXPFBKCD, respectively. Analyze the order/signal.	Your program passes a null buffer and the return-order/signal flag in the XPCCB. VSE/POWER has passed an order or a signal when the ECB is posted. Let's assume that VSE/POWER passed the device order for a PFLUSH without HOLD.
Pass an order-response record XPCC FUNC=SENDR Check the return codes as shown above. WAIT IJBXSECB Check the VSE reason codes as shown above. Check the VSE/POWER return and feedback codes as shown above.	VSE/POWER has processed the response record and returned a null buffer when the ECB is posted.
Pass a close request XPCC FUNC=SENDR Check the return codes as shown above. WAIT IJBXSECB Check the VSE reason codes as shown above. Check the VSE/POWER return and feedback codes as shown above.	Your program passes a null buffer with the XPCCB bytes set as follows: PXUACT1 to PXUATRQS PXUBTYP to zero VSE/POWER has processed the request and returned a null buffer when the ECB is posted. VSE/POWER deletes the currently processed queue entry if the entry's disposition was D. VSE/POWER retains the entry with a disposition of L, if its original disposition was K.
Get-service request for the next selectable queue entry	

Table 54. Code for a PFLUSH with HOLD Sequence

Coding in your application program	Comments
... ...	Assumption: During GET data processing the PXPIORD (order signal queued) indication is set in your program's XPCCB.
Pass a return-order/signal request XPCC FUNC=SENDR Check the return codes in register 15 and in the XPCCB (byte IJBXRETC). WAIT IJBXSECB Check the VSE reason codes in the XPCCB byte IJBXREAS. Check the VSE/POWER return and feedback codes in XPCCB bytes PXPRETCD and PXPFBKCD, respectively. Analyze the order/signal.	Your program passes a null buffer and the return-order/signal flag in the XPCCB. VSE/POWER has passed an order or a signal when the ECB is posted. Let's assume that VSE/POWER passed a device order for a PFLUSH with HOLD.
Pass an order-response record XPCC FUNC=SENDR Check the return codes as shown above. WAIT IJBXSECB Check the VSE reason codes as shown above. Check the VSE/POWER return and feedback codes as shown above.	VSE/POWER has processed the response record and returned a null buffer when the ECB is posted.
GET spool data request XPCC FUNC=SENDR Check the return codes as shown above. WAIT IJBXSECB Check the VSE reason codes as shown above. Check the VSE/POWER return and feedback codes as shown above. If the end of the current page is reached, proceed to the next step; else **return to the beginning of this step**.	When the ECB is posted, your program's reply buffer is filled with spooled output records retrieved from the accessed queue entry.

Table 54. Code for a PFLUSH with HOLD Sequence (continued)

Coding in your application program	Comments
Pass a checkpoint request XPCC FUNC=SENDR Check the return codes in register 15 and in the XPCCB (byte IJBXRETC). WAIT IJBXSECB Check the VSE reason codes in the XPCCB byte IJBXREAS. Check the VSE/POWER return and feedback codes in XPCCB bytes PXPRETCD and PXPFBKCD, respectively.	Your program passes a checkpoint-control record in its send buffer. When the ECB is posted, VSE/POWER has passed a checkpoint-response record to your program's reply buffer.
Pass a flush-hold request XPCC FUNC=SENDR Check the return codes as shown above. WAIT IJBXSECB Check the VSE reason codes as shown above. Check the VSE/POWER return and feedback codes as shown above.	Your program passes a null buffer and XPCCB bytes set as follows: PXUACT1 set to PXUATFLH PXUBTYP set to zero VSE/POWER has processed the request and returned a null buffer when the ECB is posted; processing of the affected output queue entry by VSE/POWER is canceled. The complete output queue entry is retained in its output queue with the class and priority assignments unchanged. The queue entry's disposition, however, is changed to: H if it was D. L if it was K.
Get-service request for the next selectable queue entry.	

Stopping the Device

Normally, the stopping of a device is triggered by VSE/POWER when it processes a PSTOP DEV command for the device or a PEND command.

Either command causes VSE/POWER to build a stop-device order and to add this order to the order chain for the device. The order may request the device to be stopped:

- At the end of the currently processed output

  A PSTOP command with EOJ or a PEND command was issued. Your program must provide for continued processing of output until the end of the currently processed output is reached. Table 55 on page 187 shows the coding sequence that should be followed.
- At once for restart at the point of interruption

  A PSTOP command with RESTART was issued. Your program must provide for continued processing of output until the end of a logical boundary (a page for a printer, for example) is reached. At this point, have your program request a checkpoint because setting up output processing on restart for the queue entry is your program's responsibility. Table 56 on page 189 shows the coding sequence that should be followed.

- At once for restart from the beginning

  Neither EOJ nor RESTART was specified in the PSTOP command. In this case, your program should:

  1. Purge the data that may be contained in a device buffer, if any.
  2. Issue a quit request.

Table 55. Code for Device Stop after End of Output Sequence

Coding in your application program	Comments
... ...	Assumption: During GET data processing the PXPIORD (order signal queued) indication is set in your program's XPCCB.
Pass a return-order/signal request XPCC FUNC=SENDR Check the return codes in register 15 and in the XPCCB (byte IJBXRETC). WAIT IJBXSECB Check the VSE reason codes in the XPCCB byte IJBXREAS. Check the VSE/POWER return and feedback codes in XPCCB bytes PXPRETCD and PXPFBKCD, respectively. Analyze the order/signal.	Your program passes a null buffer and the return-order/signal flag in the XPCCB. VSE/POWER has passed an order or a signal when the ECB is posted. Let's assume that VSE/POWER passed a device order for a PSTOP with EOJ.
Pass an order-response record XPCC FUNC=SENDR Check the return codes as shown above. WAIT IJBXSECB Check the VSE reason codes as shown above. Check the VSE/POWER return and feedback codes as shown above.	VSE/POWER has processed the response record and returned a null buffer when the ECB is posted.
GET spool data request XPCC FUNC=SENDR Check the return codes as shown above. WAIT IJBXSECB Check the VSE reason codes as shown above. Check the VSE/POWER return and feedback codes as shown above. If the end of the queue entry is reached, proceed; else **return to the beginning of this step**.	When the ECB is posted, your program's reply buffer is filled with output records retrieved from the accessed queue entry.

Table 55. Code for Device Stop after End of Output Sequence (continued)

Coding in your application program	Comments
Empty hardware I/O buffers	Applies if the device is buffered or connected via a communication link. Your program must ensure that records still in a hardware buffer are actually written to the device before the retrieval service for the output is closed. This avoids that VSE/POWER deletes the output before all of the output records have been transferred to and processed by the device.
Issue a CLOSE request XPCC FUNC=SENDR Check the return codes in register 15 and in the XPCCB (byte IJBXRETC). WAIT IJBXSECB Check the VSE reason codes in the XPCCB byte IJBXREAS. Check the VSE/POWER return and feedback codes in XPCCB bytes PXPRETCD and PXPFBKCD, respectively.	Your program passes a null buffer with the XPCCB bytes set as follows: PXUACT1 to PXUATRQS PXUBTYP to zero When the ECB is posted, VSE/POWER has: - returned a null buffer. - deleted the output if this output's disposition was D. - changed the output's disposition to L if this disposition was K.
Pass a device-stopped signal XPCC FUNC=SENDR Check the return codes as shown above. WAIT IJBXSECB Check the VSE reason codes as shown above. Check the VSE/POWER return and feedback codes in XPCCB bytes PXPRETCD and PXPFBKCD, respectively.	Your program passes a null buffer with XPCCB bytes set as follows: PXUSIGNL to PXUSDSTP PXUBTYP to zero VSE/POWER has processed the signal and returned a null buffer when the ECB is posted. VSE/POWER informs about the device-stopped condition by a message to the PSTART device operator and to the user who issued the PSTOP (or PEND) command, thus disconnecting the communication path.
Give up the communication path XPCC FUNC=DISCPRG Check the return codes as shown above.	The communication path is removed.

This chart shows only how a stop with a restart possibility differs from a stop after end of job.

Table 56. Code for Device Stop with a Restart Possibility Sequence

Coding in your application program	Comments
... ... Pass the required order-response record XPCC FUNC=SENDR Check the return codes in register 15 and in the XPCCB (byte IJBXRETC). WAIT IJBXSECB Check the VSE reason codes in the XPCCB byte IJBXREAS. Check the VSE/POWER return and feedback codes in XPCCB bytes PXPRETCD and PXPFBKCD, respectively.	VSE/POWER has processed the response record and returned a null buffer when the ECB is posted.
GET spool data request XPCC FUNC=SENDR Check the return codes as shown above. WAIT IJBXSECB Check the VSE reason codes as shown above. Check the VSE/POWER return and feedback codes as shown above. If the end of a logical boundary is reached, proceed; else **return to the beginning of this step**.	When the ECB is posted, your program's reply buffer is filled with output records retrieved from the accessed queue entry.
Pass a checkpoint request XPCC FUNC=SENDR Check the return codes in register 15 and in the XPCCB (byte IJBXRETC). WAIT IJBXSECB Check the VSE reason codes in the XPCCB byte IJBXREAS. Check the VSE/POWER return and feedback codes in XPCCB bytes PXPRETCD and PXPFBKCD, respectively.	Your program passes a checkpoint-control record in its send buffer. When the ECB is posted, VSE/POWER has passed a checkpoint-response record to your program's reply buffer.
Empty hardware I/O buffers	This is the same as for a termination after end of job; see the coding sequence shown in the preceding illustration.
Pass a quit request XPCC FUNC=SENDR Check the return codes as shown above.	Your program passes a null buffer and XPCCB bytes set as follows: PXUACT1 set to PXUATABR PXUBTYP set to zero

Table 56. Code for Device Stop with a Restart Possibility Sequence (continued)

Coding in your application program	Comments
WAIT IJBXSECB Check the VSE reason codes as shown above. Check the VSE/POWER return and feedback codes as shown above.	VSE/POWER has processed the request when the ECB is posted. The queue entry being processed is retained by VSE/POWER with unchanged priority and disposition assignments.
Pass a device-stopped signal	This and the remainder of the coding sequence is the same as for a termination after end of job (see in the Table 55 on page 187).

Handling an Abnormal-End Situation

An abnormal-end situation may arise

1. because of an error condition found in processing an output queue entry, or
2. because of an error condition within VSE/POWER.

Output-Related Abnormal End

This type of an abnormal-end situation may be triggered by VSE/POWER or by your program. Normally, VSE/POWER removes the communication path immediately. It writes a message to the system operator and to the device owner, retains the currently processed queue entry with its priority and disposition, and performs accounting.

If it is triggered by your program, your program should analyze the situation and do one of the following:

- Cancel itself. This action is indicated if there is no chance for continued useful work. If this occurs, VSE/POWER is informed about it by the XPCC interface.
- Remove the communication path by an XPCC request specifying FUNC=DISCPRG. This action is indicated if there is no chance for continued useful processing of data passed via the communication path to or from your program.
- Remove the communication path by an XPCC request specifying FUNC=DISCONN when the last FUNC=SENDR request has been completed (SECB posted). This action is indicated if, for example, your program can no longer write to the output device. Before removing the communication path, your program should inform the system operator and, if possible, also the device owner of the type of failure.

  If the device was active when the failure occurred, have your program save a checkpoint, a VSE/POWER-assigned record number lower than the number of the failing record. Your program can use this record number as a restart point when processing of the interrupted queue entry is resumed.

Table 57 on page 191 shows the coding sequence that should be followed when a device fails.

Table 57. Code for Abnormal End Because of a Device Failure Sequence

Coding in your application program	Comments
... ... Pass a send-message order XPCC FUNC=SENDR Check the return codes in register 15 and in the XPCCB (byte IJBXRETC). WAIT IJBXSECB Check the VSE reason codes in the XPCCB byte IJBXREAS. Check the VSE/POWER return and feedback codes in XPCCB bytes PXPRETCD and PXPFBKCD, respectively.	Tells VSE/POWER where to route the message which is part of the order control record. VSE/POWER's order response record is in your program's reply buffer when the ECB is posted.
Pass a checkpoint request XPCC FUNC=SENDR Check the return codes as shown above. WAIT IJBXSECB Check the VSE reason codes as shown above. Check the VSE/POWER return and feedback codes as shown above.	Your program passes a checkpoint-control record in its send buffer. When the ECB is posted, VSE/POWER has passed a checkpoint-response record to your program's reply buffer.
Pass a quit request XPCC FUNC=SENDR Check the return codes as shown above.	Your program passes a null buffer and XPCCB bytes set as follows: PXUACT1 set to PXUATABR PXUBTYP set to zero
WAIT IJBXSECB Check the VSE reason codes as shown above. Check the VSE/POWER return and feedback codes as shown above.	VSE/POWER has processed the request when the ECB is posted. The interrupted queue entry is retained by VSE/POWER with unchanged priority and disposition assignments.
Give up the communication path XPCC FUNC=DISCONN Check the return codes in register 15 and in the XPCCB (byte IJBXRETC).	The communication path is removed.

In case of an output-processing failure indicated to your program, you can also issue a ‘quit-and-lock’ request at any point during the retrieval of a queue entry. The request causes VSE/POWER to re-queue the currently processed queue entry in the appropriate class chain with a temporary disposition of Y for the purpose of:

- Indicating that a problem has occurred during output processing, and
- Preventing that the output queue entry is handled again until the subsystem has taken some action.

For further information, see “Issuing a QUIT-and-LOCK Request” on page 87.

Whenever the communication path is removed before output processing for a 'protected' queue entry could be terminated by any GET end-service request, VSE/POWER requeues the output entry with a temporary disposition Y. For creation of 'protected' queue entries see "Handling an Abnormal-End Condition During GET" on page 100.

Abnormal End of VSE/POWER

VSE/POWER itself may happen to be canceled during output processing. The XPCC interface informs your program about this by passing to your program XPCCB return or reason codes of IJBXNOC3 and IJBXABDC, respectively. Your program can, in this case:

1. Empty hardware-output buffers, if any.
2. When VSE/POWER is up again, restart the interrupted processing either:
   - At a suitable checkpoint (if the output was checkpointed). Obtaining checkpoints during data retrieval is described under "Requesting a Checkpoint" on page 88; restarting at a checkpoint is discussed in the section "Requesting a Restart of the GET Spool Data" on page 93.
   - At the beginning of the interrupted output.

For more information about the retrieval and restart of a queue entry, see Chapter 8, "GET - Retrieving a Queue Entry," on page 75.

If VSE/POWER or the XPCC interface happens to be canceled while processing a 'protected' output queue entry, VSE/POWER recovery (at system warm start) or the VSE/POWER device-service task will re-queue the output entry with disposition Y to the non-dispatchable queue. For creation of a protected queue entry see "Handling an Abnormal-End Condition During GET" on page 100.

Processing of Order-Control Records and Signals

Orders and signals are used to synchronize a VSE/POWER device-service task with your program.

An order is a control record which is passed from one side of a communication path to the other. A signal is a status indication that is passed to the other end of the communication path. Orders that VSE/POWER can pass to your program are referred to as device orders; orders that your program can pass to VSE/POWER are called subsystem orders.

VSE/POWER-Built Device Orders

VSE/POWER builds a device order whenever it processes any of the following commands for a device under your program's control:

```
Command     Order-Type
PSTART      Start-device order
PSTOP       Stop-device order
PRESTART    Restart-device order
PGO         Reactivate-device order
PSETUP      Setup-device order
PFLUSH      Cancel-output device order
PXMIT       Transmit-command device order
```

VSE/POWER handles device orders in a first-in first-out way by chaining them, one behind the other, separately for every device controlled by your program. VSE/POWER accepts a command for a device even after a PSTOP DEV command was processed for this device, that is, until your program has passed a device-stop signal.

Subsystem-Originated Orders

The subsystem (your program) would build an order and pass it to VSE/POWER whenever the need arises. Your program can build and pass orders of the following type:

Send-message order.

Set-logical-destination order.

Put-account record order.

Process a Device Order

Process Overview

When having passed a device order to your program, VSE/POWER expects that the program analyses the order immediately and returns a corresponding order-response record. If your program fails to return this record, VSE/POWER discontinues the communication path and informs your program by a return code of PXPRCPVL together with the applicable feedback code. VSE/POWER stops the communication path also if your program's order-response record does not correspond to the type of order passed by VSE/POWER.

The order-response record shows your program's decision: accepted or not accepted. If the decision is not accepted, the record may also indicate a reason for rejecting the device order; it may include a message for VSE/POWER to route to the user whose command triggered the device order. For the programmed actions that are to be coded to return an order-response record, see Table 58 on page 194.

A message generated by VSE/POWER in response to a command is routed to the command originator whose node ID and user ID may be derived from the device-order header. A message passed to VSE/POWER as part of an order-response control record is routed to the user indicated in this record; by default, this is the originator of the command. For details on message routing, refer to Table 60 on page 197.

A device order, once accepted by an order-response record, may be processed by your program some time later. For example, after having accepted an immediate-stop device order, your program can request a checkpoint to be taken before it processes the order. There is one exception, however: the start-device order. Your program must process this order immediately and return the result of this processing by way of an order-response record valid for this device order.

Sequence of Events

1. VSE/POWER chains a device order for being passed via a communication path when it processes a command for the involved device. This may occur at any time. VSE/POWER indicates the chaining of a device order.
2. VSE/POWER indicates the chaining of a device order by setting the order-pending flag in the XPCCB for the communication path. When this XPCCB is passed to the other end (your program), VSE/POWER expects, sooner or later, a return-order/signal request to be returned. In short, your program should be ready to pick up and analyze a device order every time VSE/POWER has passed to your program a block of output records.
3. In response to a return-order/signal request, VSE/POWER passes the device order at the head of the chain if two or more such orders are chained for the

communication path. The order-pending flag remains set as long as a device order waits for being passed to your program.

Table 58 shows the coding sequence which you should follow in your program for the handling of device orders.

Size of Your Reply Buffer

An order-control record can be up to 180 bytes long. Therefore, the size of your program's reply buffer should be 180 bytes or larger.

Table 58. Code for Processing of Device Orders Sequence

Coding in your application program	Comments
... ... GET spool data request for the next block XPCC FUNC=SENDR Check the return codes in register 15 and in the XPCCB (byte IJBXRETC). WAIT IJBXSECB Check the VSE reason codes in the XPCCB byte IJBXREAS. Check the return-and-feedback codes in XPCCB bytes PXPRETCD and PXPFBKCD, respectively. Check the XPCCB byte PXPINFO.	When the ECB is posted, your program's reply buffer is filled with spooled output records retrieved from the accessed queue entry. An order or signal is chained if the PXPIORD bit of this byte is set on.
Process the data passed by VSE/POWER	Prepare this data for writing it to the involved output device.
Pass a return-order/signal request XPCC FUNC=SENDR Check the return codes as shown above.	Your program passes a null buffer with XPCCB bytes set as follows: PXUBTYP to zero PXUACT1 to PXUATROR
WAIT IJBXSECB Check the VSE reason codes as shown above. Check the VSE/POWER return and feedback codes as shown above. Analyze the order.	When the ECB is posted, VSE/POWER has passed an order or a signal, if there was one; if there was none, VSE/POWER indicates this by a return and feedback code combination of PXPRCOKF and PXP04NOQ. Let's assume that VSE/POWER passed a device order.

Table 58. Code for Processing of Device Orders Sequence (continued)

Coding in your application program	Comments
Pass the required order-response record XPCC FUNC=SENDR Check the return codes in register 15 and in the XPCCB (byte IJBXRETC).	With only the control record in your program's send buffer and with the XPCCB's byte PXUBTYP set to PXUBTCTL.
WAIT IJBXSECB Check the VSE reason codes in the XPCCB byte IJBXREAS. Check the return-and-feedback codes in XPCCB bytes PXPRETCD and PXPFBKCD, respectively.	VSE/POWER has processed the response record and returned a null buffer when the ECB is posted.

Process a Subsystem Order

To pass an order to VSE/POWER, your program must:

1. Set up the device order as the only data in the communication path's send buffer.
2. Issue an XPCC request specifying FUNC=SENDR. The XPCCB used for the request must have its user-information byte PXUBTYP set to PXUBTCTL.

VSE/POWER analyzes the order and returns to your program the corresponding order-response record. For information about the format and contents of the orders and response records, see "Device/Subsystem Orders and Order-Response Records" below.

Device/Subsystem Orders and Order-Response Records

Device/subsystem orders and order-response records are similar in format. Both types of control records have a header section and a variable-data section. Following below are:

1. The format and description of the header section of a device/subsystem order. The description includes a general discussion of the data section; the required details about order data sections are given separately by device/subsystem orders.
2. The format and description of the order-response record, including its data section.

Device/Subsystem-Order Header Section

For the format of this record section and a discussion of its contents, refer to Table 59. In the assembly output listing for the PWRSPL macro with TYPE=MAP, you find a DSECT for the record section at the label PORDER.

Table 59. Device/Subsystem-Order Header Section Format

Bytes	Field	Contents / Description
0-1	PORDRLEN	Record length (in binary notation).
2	PORDTYPE	X'05' – Device-order indicator.
3	PORDMOD	Device-order type:

Table 59. Device/Subsystem-Order Header Section Format (continued)

Bytes	Field	Contents / Description			
		Mnemonic	**Value**	**Order-type**	**Triggered by**
		PORDMSTR	X'01'	Start device	PSTART
		PORDMSTP	X'02'	Stop device	PSTOP
		PORDMRST	X'03'	Restart device	PRESTART
		PORDMPGO	X'04'	Reactivate device	PGO
		PORDMSET	X'05'	Setup device	PSETUP
		PORDMFLH	X'06'	Cancel processing	PFLUSH
		PORDMXMT	X'07'	User defined	PXMIT
		PORDMSND	X'10'	Send message	Subsystem
		PORDMSLD	X'11'	Set logical destination	Subsystem
		PORDMPAO	X'12'	Put account record	Subsystem
4	PORDFLAG	Flag byte. To be set to X'80' by the subsystem in a send-message order if the message is to be held for redisplay (by a PDISPLAY M command).			
5	PORDMSGL	Length of message (in binary notation). To be supplied by the subsystem in a send-message order.			
6-7	PORDAFPL	Length of Advanced Function Printing account record. To be supplied by the subsystem in a PUT-account record order.			
8-F	PORDSUBS	Requesting[1] subsystem's name (in character notation).			
10-17	PORDNODE	Requesting[1] node's name (in character notation). Your own z/VSE system's node name (or blank) if the triggering command was submitted within the domain of your node.			
18-1F	PORDUSER	Requesting:sup.1:esup. user's ID (in character notation) Blank if the command was entered by a central operator.			
20-n		Variable-data area. See also the Note below.			

[1] If send-message order, the field contains the destination information instead of the requestor's information.

Note: Details are given in the sections discussing the device/subsystem orders. The variable-data area includes a parameter string if one was specified in the triggering command. This string normally provides operator-specified information that your program needs. Tell your operator what to specify and how.

VSE/POWER's requirements regarding the parameter string are:

- It may not be longer than 60 characters. This includes blanks or commas that your program may need as delimiters.
- It must start with an alphameric character in the first character position.
- It must include at least one blank in any of the second through 16th character positions.
- An apostrophe (') within the string must be entered by the operator as two apostrophes ('').

Order-Response Record

When VSE/POWER passes a device order, it expects your program to return the corresponding order-response record with your program's next XPCC request. If your program passes an invalid response record, VSE/POWER:

1. Rejects this record with a return/feedback-code combination of PXPRCERR/PXPO8UXR in the XPCCB bytes PXPRETCD and PXPFBKCD.
2. Waits for a new corrected response record.

When your program passes a subsystem order, VSE/POWER returns the corresponding order-response record also in response to the next XPCC request.

For the format of the record and a discussion of its contents refer to Table 60. In the assembly output listing for the PWRSPL macro with TYPE=MAP, you find a DSECT for the record section at the label PORDRESP.

Table 60. Order-Response Control Record Format

<table>
<tr><th>Bytes</th><th>Field</th><th colspan="3">Contents / Description</th></tr>
<tr><td>0-1</td><td>PORSRLEN</td><td colspan="3">Record length (in binary notation).</td></tr>
<tr><td>2</td><td>PORSTYPE</td><td colspan="3">X'06' - Order-response record indicator.</td></tr>
<tr><td>3</td><td>PORSMOD</td><td colspan="3">Device-order type – The type indicator of the device order to which a response is being made. Consider picking up field PORDMOD of the device order, which is discussed under “Device/Subsystem-Order Header Section” on page 195.</td></tr>
<tr><td>4</td><td>PORSFLAG</td><td colspan="3">Flag byte:</td></tr>
<tr><td></td><td></td><td>Mnemonic</td><td>Value</td><td>Meaning</td></tr>
<tr><td></td><td></td><td>PORSFMID</td><td>X'80'</td><td>PORSMID contains 4-byte message-id.</td></tr>
<tr><td></td><td></td><td colspan="3">Following a send-message order, once the message has been issued to the local or central operator, then the system returns the message-id with which the subsystem can delete the message from the console screen using the DOM macro. This is necessary for 'highlighted' action messages (for example, MOUNT FORMS) that the operator would otherwise have to delete manually.</td></tr>
<tr><td>5</td><td>PORSMSGL</td><td colspan="3">Length of the message (in binary notation), if there is one; else X'00'.</td></tr>
<tr><td>6</td><td>PORSRETC</td><td colspan="3">Order return code:</td></tr>
<tr><td></td><td></td><td>Mnemonic</td><td>Value</td><td>Meaning</td></tr>
<tr><td></td><td></td><td>PORSROK</td><td>X'00'</td><td>Order accepted.</td></tr>
<tr><td></td><td></td><td>PORSROKF</td><td>X'04'</td><td>Order accepted; unable to handle request.</td></tr>
<tr><td></td><td></td><td>PORSRINV</td><td>X'08'</td><td>Order not accepted.</td></tr>
<tr><td>7</td><td>PORSFDBK</td><td colspan="3">Order feedback code:</td></tr>
</table>

Table 60. Order-Response Control Record Format (continued)

Bytes	Field	Contents / Description		
		Mnemonic	**Value**	**Meaning**
		From the subsystem to VSE/POWER:		
		PORSFOK	X'00'	All OK.
		PORSFPAR	X'01'	Missing or invalid parameter string.
		PORSFONA	X'02'	Subsystem-internal reason.
		PORSFDUN	X'03'	Device to be started is unknown.
		PORSFDBS	X'04'	Device to be started is busy.
		PORSFDOS	X'05'	Device to be started out of service.
		PORSFDRJ	X'06'	Device start rejected for subsystem internal reason
		From VSE/POWER to the subsystem:		
		PORSFNAC	X'01'	Accounting support not initialized.
		PORSFINV	X'01'	Order is invalid or unknown.
		PORSFOTS	X'02'	Order is too short versus contents.
		PORSFMSG	X'03'	Message text is too long.
		PORSFSLD	X'04'	Invalid destination in a preceding set-logical destination order.
		PORSFPAC	X'05'	The passed order length is not equal to the length of the order header plus specified length of the account record.
		PORSFRTL	X'06'	The passed account record is either too small (less than 55 bytes) or too large (larger than 1000 bytes).
8-F	PORDSUBS	Destination subsystem's name (in character notation).		
10-17	PORDNODE	Destination node's name (in character notation). Blank if the message passed with the order-response record is to be routed to the system operator.		
18-1F	PORDUSER	Destination user's ID (in character notation). Blank if the message passed with the order-response record is to be routed to the system operator.		
20-97	PORSMSG	Message text[1]		
20-23	PORSMID	Message-id if flag PORSFMID is set and VSE/POWER returns an order-response record.		

Table 60. Order-Response Control Record Format (continued)

Bytes	Field	Contents / Description
[1] Applies to order-response records from the subsystem to VSE/POWER. The content of the field PORSMSG, the message text, is picked up by VSE/POWER. It must be alphanumeric and can be up to 120 characters long. A shorter text must be padded with trailing blanks. Your program can include an error message here if, for example, the parameter string passed with the device order is in error. VSE/POWER routes this message to the user identified by fields PORDSUBS, PORDNODE, PORDUSER, and translates the message text to uppercase. For more details on how the message is displayed on the central operator console, see "Send-Message Order" on page 205.		

Start-Device Order

VSE/POWER passes the order to your program when a PSTART DEV command is processed for a device under your program's control. Not until it has accepted the order (by a corresponding order-response record) can your program request VSE/POWER to pass output spooled for the device.

If the device cannot be started, your program must indicate this and give a reason by setting the return-and-feedback codes in the order-response record. A return code other than PORSROK (X'00') causes VSE/POWER to discontinue the communication path.

Table 61 and Table 62 show the format of the device order's data section and the return-and-feedback codes that your program may have to supply in the response record.

Table 61. Start-Device Order: Data Section

Bytes	Field	Contents/Description		
20 - 27	PORDSDEV	Device name specified in the PSTART command		
28 - 2B	PORDSCLS	Class(es) specified in the PSTART command		
2C - 2D		Reserved for future use		
2E	PORDSFLG	Flag byte:		
		Mnemonic	**Value**	**Meaning**
		PORDSSKP	X'80'	PSTART with SKIP=YES
2F	PORDSPSL	Length of parameter string (binary)		
30 - 6B	-	Parameter string as supplied in the PSTART command		

Table 62. Start-Device Order: Response-Record Return and Feedback Codes

Return Code		Feedback Code		
Mnemonic	**Value**	**Mnemonic**	**Value**	**Meaning**
PORSROK	X'00'	PORSFOK	X'00	Order accepted, device started
PORSRINV	X'08'			Order not accepted
		PORSFPAR	X'01'	Parameter string missing or invalid.
		PORSFONA	X'02'	Subsystem-internal reason
		PORSFDUN	X'03'	Device to be started is unknown.
		PORSFDBS	X'04'	Device to be started is busy.
		PORSFDOS	X'05'	Device to be started out of service.

Table 62. Start-Device Order: Response-Record Return and Feedback Codes (continued)

Return Code		Feedback Code		
Mnemonic	Value	Mnemonic	Value	Meaning
		PORSFDRJ	X'06'	Device start rejected for subsystem internal reason.

Stop-Device Order

Table 63 and Table 64 show the format of the device order's data section and the return-and-feedback codes that your program may have to supply in the response record.

VSE/POWER passes the order to your program when either of the following occurs:

- A PSTOP DEV command is processed for a device under your program's control.
- An orderly VSE/POWER shutdown in response to a PEND command is in process.

VSE/POWER honors your program's GET-spooled data requests even after the program has passed the corresponding response record. In fact, it honors these requests until your program has passed its device-stopped signal.

Table 63. Stop-Device Order: Data Section

Bytes	Field	Contents/Description		
20	PORDPTRB	Termination request byte:		
		Mnemonic	Value	Meaning
		PORDPEOJ	X'80'	Stop at end of job
		PORDPIMM	X'40'	Stop immediately
		PORDPRST	X'20'	Stop for later restart
21-22		Reserved		
23	PORDPPSL	Length of parameter string		
24-5F	PORDPPRM	Parameter string		

Table 64. Stop-Device Order: Response-Record Return and Feedback Codes

Return Code		Feedback Code		
Mnemonic	Value	Mnemonic	Value	Meaning
PORSROK	X'00'	PORSFOK	X'00	Order accepted, device started
PORSRINV	X'08'			Order not accepted
		PORSFPAR	X'01'	Parameter string missing or invalid.
		PORSFONA	X'02'	Subsystem-internal reason

A PSTOP DEV,..,FORCE command does not cause a stop-device order to be passed by VSE/POWER. Instead, VSE/POWER discontinues the communication path immediately. VSE/POWER informs your program about this by a return- and feedback-code combination of PXPRCNOC and PXP10PSP.

Setup-Device Order

VSE/POWER passes the order to your program when a PSETUP DEV command is processed for a device under your program's control. The order indicates the number of pages that are to be printed so that the operator can do the required device setup. As a help for the device operator, consider having your program replace on the setup pages:

all letters by the character X

every digit of a number by a 9

Table 65 and Table 66 below show the format of the device order's data section and the return-and-feedback codes that your program may have to supply in the response record.

Table 65. Setup-Device Order: Data Section

Bytes	Field	Contents/Description
20-23	PORDUPGE	Number of pages (in binary notation).
24-2E		Reserved
2F	PORDUPSL	Length of parameter string (in binary notation)
30-6B	PORDUPRM	Parameter string

Table 66. Setup-Device Order: Response-Record Return and Feedback Codes

Return Code		Feedback Code		
Mnemonic	**Value**	**Mnemonic**	**Value**	**Meaning**
PORSROK	X'00'	PORSFOK	X'00	Order accepted, device started
PORSRINV	X'08'			Order not accepted
		PORSFPAR	X'01'	Parameter string missing or invalid.
		PORSFONA	X'02'	Subsystem-internal reason

Your program must inform VSE/POWER when it is finished with the setup processing. This is done by passing a setup-processed signal. The signal causes VSE/POWER to re-position its retrieval pointers to the beginning of the currently processed queue entry.

Reactivate-Device Order

VSE/POWER passes the order to your program when a PGO DEV command is processed for a device under your program's control. Table 67 and Table 68 on page 202 show the format of the device order's data section and the return-and-feedback codes that your program may have to supply in the response record.

Table 67. Setup-Device Order: Data Section

Bytes	Field	Contents/Description
20-22		Reserved
23	PORDGPSL	Length of parameter string
24-5F	PORDGPRM	Parameter string

Table 68. Setup-Device Order: Response-Record Return and Feedback Codes

Return Code		Feedback Code		
Mnemonic	**Value**	**Mnemonic**	**Value**	**Meaning**
PORSROK	X'00'	PORSFOK	X'00	Order accepted, device started
PORSRINV	X'08'			Order not accepted
		PORSFPAR	X'01'	Parameter string missing or invalid.
		PORSFONA	X'02'	Subsystem-internal reason

Restart-Device Order

VSE/POWER passes the order to your program when a PRESTART DEV command is processed for a device under your program's control. Table 69 and Table 70 show the format of the device order's data section and the return-and-feedback codes that your program may have to supply in the response record.

Table 69. Restart-Device Order: Data Section

<table>
<tr><th>Bytes</th><th>Field</th><th colspan="3">Contents / Description</th></tr>
<tr><td>20</td><td>PORDTFLG</td><td colspan="3">Restart-sign flag:</td></tr>
<tr><td rowspan="4"></td><td rowspan="4"></td><td>Mnemonic</td><td>Value</td><td>Meaning</td></tr>
<tr><td>PORDTPOS</td><td>X'80'</td><td>Plus sign (forward count)</td></tr>
<tr><td>PORDTMIN</td><td>X'40'</td><td>Minus sign (backward count)</td></tr>
<tr><td>PORDTABS</td><td>X'20'</td><td>No sign (start from the beginning)</td></tr>
<tr><td>21 - 23</td><td></td><td colspan="3">Reserved</td></tr>
<tr><td>24 - 27</td><td>PORDTPGE</td><td colspan="3">Number of pages/printlines &bxh. How to interpret this number depends on your application.</td></tr>
<tr><td>28 - 2E</td><td></td><td colspan="3">Reserved</td></tr>
<tr><td>2F</td><td>PORDTPSL</td><td colspan="3">Length of parameter string</td></tr>
<tr><td>30 - 6B</td><td>PORDTPRM</td><td colspan="3">Reserved</td></tr>
</table>

Table 70. Restart-Device Order: Response-Record Return and Feedback Codes

Return Code		Feedback Code		
Mnemonic	**Value**	**Mnemonic**	**Value**	**Meaning**
PORSROK	X'00'	PORSFOK	X'00	Order accepted, device started
PORSRINV	X'08'			Order not accepted
		PORSFPAR	X'01'	Parameter string missing or invalid.
		PORSFONA	X'02'	Subsystem-internal reason

Cancel-Output Order

VSE/POWER passes the order to your program when a PFLUSH DEV command is processed for a device under your program's control.

Table 71 and Table 72 show the format of the device order's data section and the return-and-feedback codes that your program may have to supply in the response record.

Table 71. Cancel-Output Order: Data Section

Bytes	Field	Contents/Description
20	PORDFFLG	HOLD indicator – HOLD was specified in the command if the byte is set to PORDFHLD (X'80'); else, the byte is set to X'00'.
21-22		Reserved
23	PORDFPSL	Length of parameter string
24-5F	PORDFPRM	Parameter string

Table 72. Cancel-Output Order: Response-Record Return and Feedback Codes

Return Code		Feedback Code		
Mnemonic	Value	Mnemonic	Value	Meaning
PORSROK	X'00'	PORSFOK	X'00	Order accepted, device started
PORSRINV	X'08'			Order not accepted
		PORSFPAR	X'01'	Parameter string missing or invalid.
		PORSFONA	X'02'	Subsystem-internal reason

Transmit-Command Order

VSE/POWER passes the order to your program when a PXMIT DEV command is processed for a device under your program's control. The command specified in the PXMIT command is passed to your program unchanged.

Table 73 and Table 74 show the format of the device order's data section and the return-and-feedback codes that your program may have to supply in the response record.

Table 73. Transmit-Command Order: Data Section

Bytes	Field	Contents/Description
20	PORDXPSL	Length of the specified command
21-A4	PORDXPRM	The command specified in the PXMIT command

Table 74. Transmit-Command Order: Response-Record Return and Feedback Codes

Return Code		Feedback Code		
Mnemonic	Value	Mnemonic	Value	Meaning
PORSROK	X'00'	PORSFOK	X'00	Order accepted, device started
PORSRINV	X'08'			Order not accepted
		PORSFONA	X'02'	Subsystem-internal reason

Subsystem Orders

VSE/POWER accepts and processes subsystem orders as follows:
- Send-message order
- Set-logical-destination order
- Put-account record order

To pass an order to VSE/POWER, your program must:
1. Set the buffer-type flag in the XPCCB to indicate that your program's send buffer contains a control record.
2. Ensure that the buffer contains the correct order-control record and nothing else.
3. Issue an XPCC request with FUNC=SENDR.

Your program can pass an order at any time after completion of a preceding request.

VSE/POWER replies to the order with the corresponding response record. Table 75 shows the return-and-feedback codes which VSE/POWER may set in its response record.

Table 75. Subsystem Orders: Response Codes

Return Code		Feedback Code		
Mnemonic	**Value**	**Mnemonic**	**Value**	**Meaning**
PORSROK	X'00'	PORSFOK	X'00	Order accepted
PORSROKF	X'04'	PORSFNAK	X'01'	Accounting function not initialized
PORSRINV	X'08'			Order not accepted
		PORSFINV	X'01'	Order is invalid or unknown
		PORSFOTS	X'02'	Order is too short
		PORSFMSG	X'03'	Message text is too long
		PORSFSLD	X'04'	Invalid destination in a preceding set-logical destination order
		PORSFPAC	X'05'	Length fields mismatch with order record
		PORSFRTL	X'06'	Account record is either too small (< 55 bytes) or too large (> 1,000 bytes)

Note: The order response comes from the local system even if the order was routed to another NODEID (for example, send-message order).

Send-Message Order

Your program would pass a send-message order when it detects an error or an intervention-required condition on the involved device. This order includes the message that your program wants to be routed to the responsible operator or user.

VSE/POWER routes the message as instructed – to the system console if the order does not include a user ID. It issues the message with all alphabetic characters converted to uppercase.

A message directed to the system console is preceded by the VSE/POWER provided header: "From device:". However, selected action type messages of the CICS Report Controller and all action type messages of the Print Services Facility™ (PSF) are headed by "1QZ2A" so that they will not scroll off the console screen. All other PSF messages are headed by "1QZ2I".

If the message cannot be forwarded to its final destination, then VSE/POWER discards the message without informing your program. Therefore, if your program requires a reply to the message, be sure to supply the ID of a user that you know to be online.

The data section of a send-message order (labeled PORDMMSG) contains the free-format message as set up by your program. This message can be up to 120 alphameric characters long.

Set-Logical-Destination Order

A user can route a job's output to a certain destination. This is done by specifying, in an * $$ LST (* $$ PUN) statement for the output, a user ID with or without a node name. If this ID is the name of a device under your program's control, then the output is selectable for processing by your program.

By way of a set-logical-destination order, you can instruct VSE/POWER to “equate” up to eight names to the one by which the involved output device is known in your program. VSE/POWER then selects an output for processing by this device if it is destined for an equated user.

However, if the original name by which the output device is known in your program is to be used as user ID for routing output further, that name must be included in the list of logical destinations. An operator who issued a PSTART DEV command for a device can control that device only by commands using the same device name.

You may define identical logical destinations for several (or all) devices used under your program's control for the processing of spooled output. If you do this, two or more of these devices are available for the processing of output for certain logical destinations. In other words, you get a certain pool effect for your output devices. Consider this if you see a need for load levelling for the involved output devices.

You can pass a set-logical-destination order for a device at any time after this device has been started in response to a start-device order.

VSE/POWER uses the defined logical destination names when your program passes the next generic GET-OPEN service request via the same communication path. Therefore, code a set-logical-destination order followed by a generic GET-OPEN service request at the point where your program finds VSE/POWER's service task waiting for work. The set-logical-destination order may make one or more output queue entries selectable for processing by your program.

In a set-logical-destination order, bytes 0 through 3 of the *header section* are used as shown in Table 59 on page 195; the remaining bytes of this section are of no significance. The order's *data section*, an area of 64 bytes at label PORDDLOG, is used for the definition of logical destinations, names of up to eight alphameric characters, as follows:

1. Fill the area with blanks.
2. Do not use R000 thru R250 as logical destination, since these are reserved for RJE userid's and will lead to a rejection of the subsystem order with return error PORSRETC/PORSFDBK=X'08/04'.
3. Specify the destination names, one after the other and one per name slot of eight bytes. Include the logical name of the output device, if necessary. For VSE/POWER, a blank in the first character position of a name slot means that no more names follow.

Put-Account Record Order

The VSE/POWER spool-access-operation account record (for layout, see Table 15 on page 26) written by VSE/POWER when your program has received an output entry via a GET request, may not contain accurate page or copy counts, because VSE/POWER does not interpret Advanced Function Printing related information of the report entry. To allow for accurate printing charges, you can define your own Advanced Function Printing account record, make it part of an account record order, and ask VSE/POWER to write your account record order to the VSE/POWER account file.

The PUT-account record order can be sent at any time for a processed output entry, even when the entry has been deleted from the VSE/POWER output queue in between. To identify the output queue entry uniquely, your private account record must:

- Start with the standard VSE/POWER account record header filled by your program with information according to the layout given on Table 4 on page 11.
- Fill your own Advanced Function Printing account-record information into the area beginning at label ACAFPBDY (see Table 4 on page 11).
- Provide a layout description for this area, so that accounting evaluation programs may interpret your data.

The following fields within the order header, as shown in Table 59 on page 195, must be provided to make up a PUT-Account-Record Order:

PORDRLEN
offering the order header length plus the length of the appended account record

PORDTYPE
saying by X'05'= this is an order control record

PORDMOD
identifying by PORDMPAO (X'12'): this is an account order

PORDAFPL
offering the length of the account record which starts at label PORDAFPA.

VSE/POWER requires a minimum length of 55 bytes:

45 bytes standard account-record header

2 bytes length field ACAFPLEN

8 bytes to identify the order originator in field ACAPPLID

and does **not** allow account records longer than 1,000 bytes.

When VSE/POWER has accepted your PUT-account-record order and has written the passed account record to the VSE/POWER account file, an order response control record is returned to your program accompanied by the user data return feedback code combination PXPRCOK/PXP00OK=00/00. Then VSE/POWER has updated the following fields of the account record:

ACIDEN
: set to 'A'= AFP account record.

ACAFPLEN
: set to the value of PORDAFPL which specifies the total length of the account record.

ACAPPLID
: set to the XPCC application-id of your program.

If your passed account record can not be written to the account file due to a full condition, the control operator is informed to save or empty the account file. During this period, the SENDR request of your program does not complete and no other request may be passed until your program has received the account-record order response.

VSE/POWER may signal the following failures with the user data return/feedback code PXPRCERR/PXP08IOR=08/30 that accompany the order response record where more detailed failure reason are given as shown in Table 75 on page 204.

PORSROKF/PORSFNAC=04/01
: VSE/POWER has been started without accounting support

PORSRINV/PORSFOTS=08/02
: Your passed order control record is longer than the XPCC SENDR length

PORSRINV/PORSFPAC=08/05
: PORDRLEN does not provide a length of an order header plus the length of your account record (given in PORDAFPL)

PORSRINV/PORSFRTL=08/06
: PORDAFPL specifies an account record that is either too short (< 55 bytes) or too long (> 1,000 bytes).

Process a Signal

Signals supply status information required at the other end of a communication path. VSE/POWER and your program can work with status signals as follows:

- Output-arrived signal

  VSE/POWER passes this signal to your program when an output queue entry has become available for processing on the involved device. If you operate in a shared-spooling environment, this output may have been placed into the output queue by one of the other sharing systems.

  The format of this signal, a control record, is shown in Table 76 on page 208. VSE/POWER passes the record as the only one to your program's reply buffer for the communication path after a wait-for-order/signal or return-order/signal request. VSE/POWER needs no specific response after having passed an output-arrived signal.

  Note: A generic GET-OPEN request in response to an output-arrived signal may nevertheless result in a "no entry available" response by VSE/POWER. Another user of your system may have requested that this selectable output queue entry be processed, or the entry's class may have changed.

- Device-stopped signal

  VSE/POWER expects this signal from your program after (but not necessarily in immediate response to) a stop-device order. Your program should pass the signal to VSE/POWER after all available records have been processed on the involved device.

Table 76. Output-Arrived Signal Control Record

Bytes	Field	Contents / Description
0-1	PSGNRLEN	Record length.
2	PSGNLTYP	X'07' - Signal-control record indicator.
3	PSGNLMOD	X'01' - Output-arrived indicator.
4-7		Reserved

- Setup-processed signal

  VSE/POWER expects this signal from your program after (but not necessarily in immediate response to) a setup-device order. Your program should pass the signal to VSE/POWER when the program's processing for the necessary setup activity is complete.

To pass a signal to VSE/POWER, your program must:

1. Set up a null buffer (set IJBXBLN to zero).
2. Set byte PXUBTYP of the XPCCB to zero.
3. Set byte PXUSIGNL of the XPCCB to PXUSDSTP (for device-stopped) or PXUSSET (for setup processed).
4. Issue an XPCC request specifying FUNC=SENDR.
5. Check the return codes in register 15 and in the XPCCB byte IJBXRETC.
6. Issue a WAIT IJBSECB.
7. When the ECB is posted, VSE/POWER has returned a null buffer and passed return/feedback codes in the XPCCB user data. Check the VSE reason code in field IJBXREAS, and the VSE/POWER return-and-feedback codes.

General Hints

The following remarks generally apply to using the external device support.

Routing of VSE/POWER-Generated Messages for External Devices

If the device owner issuing the PSTART DEV,devname command is not the local central operator but, for example, a remote-node operator (or an authorized subsystem administrator), then VSE/POWER routes all messages concerning the device status to

1. The *device owner* (PSTART DEV operator), **and** to
2. The *central operator*, if required by the severity of the message, or even to
3. The *command originator*, if DEV-type commands for an already started output device originated from a third party.

Range of Support for Communicating with a Subsystem

Throughout the preceding discussion of the external device support it was assumed that, to process an output queue entry, your program would normally issue a generic GET service request with PWRSPL...QUEUE=LST specified. It is also possible to issue a

- generic GET service request to the PUN queue
- direct (specific) GET service request to the LST/PUN queue
- (specific) GET service request to the LST/PUN queue but with limited return and feedback code information
- CTL service request to any of the VSE/POWER queues.

GET requests to the RDR/XMT queue and PUT requests are not allowed.

No password checking is performed for a queue entry that is to be processed under the subsystem control.

Use of VSE/POWER Commands During Program Debug Activities

As a help in program debugging, you can consult the output as displayed by the following commands:

- PDISPLAY A,DEV
- PINQUIRE ALL|DEV|DEV=devname

For both commands, see the examples in *VSE/POWER Administration and Operation*, SC34-2625, following the description of the respective commands.

Use the PSTOP DEV,devname,FORCE command if you want to force an immediate termination of the communication path to a subsystem device.

Chapter 12. Spool-Access Support Macros

For each of the described macros, the information is given in applicable sections as follows:

1. A short summary of the macro's purpose.
2. The macro's format as used for access to VSE/POWER services.
3. A description of the macro's operands.
4. Possible return codes

For further detail on the z/VSE macros MAPXPCCB, XPCC, and XPCCB, see *z/VSE System Macros User's Guide*, SC33-8407 and *z/VSE System Macros Reference*, SC34-2638. These publications give a complete description of the macros' function and return codes.

The following chapter describes only a subset of these macros' functions. Likewise, only those functions are used in the examples that are pertinent to an understanding of the spool-access support.

Note that you must use the SENDR function (send with reply) to communicate with VSE/POWER. But you may consider making use of the 31-bit addressing support of the XPCC macro.

For an explanation of the syntax, see Chapter 1, "Understanding Syntax Diagrams," on page 3. Continuation codes that may be required in column 72 are not shown as part of the macro formats.

XPCCB

The macro sets up a cross-partition control block. Logically, the block represents one communication path. For a full description of the XPCCB macro, consult the *z/VSE System Macros Reference*, SC34-2638 publication.

Format of the Macro

```
►►──┬──────┬──XPCCB APPL=name,TOAPPL=──┬─ANY────┬──────────────────────────►
    └─name─┘                           └─SYSPWR─┘

►──┬──────────────────────────────────────┬──┬──────────────────────────────┬──►◄
   └─,BUFFER=──┬─(buffname,length)─┬──────┘  └─,REPAREA=(areaname,length)──┘
               └─bflstadr──────────┘
```

Required RMODE: **24** or **ANY**

APPL=name
: For `name` specify the name of your program. The characters SYS as the first three characters of a name are reserved for IBM subsystems.

TOAPPL=ANY|SYSPWR
: Specify:

TOAPPL=ANY
If your application makes use of the external device support (for more detail see Chapter 11, “Supporting I/O Devices Via Device Driving Systems,” on page 171).

TOAPPL=SYSPWR
If your application accesses VSE/POWER services for queue manipulation and for the retrieval or submission of jobs and output.

BUFFER=(buffname,length)|bflstadr
In the operand, `buffname` is the name of your program's send buffer.

For `length` specify the buffer's length in number of bytes. For the transfer of data to VSE/POWER, this buffer may be up to 65,535 bytes long.

If you do not specify a length, your program must insert this length into field IJBXBLN of the XPCCB.

If your program's send buffer is concatenated from several buffer segments, specify `bflstadr`; it should be the address of a list of 8-byte segment description fields as described under 'BUFFER parameter' in the XPCC macro; refer to “XPCC.”

REPAREA=(areaname,length)
In the operand, `areaname` is the name of your program's reply buffer.

For `length` specify the buffer's length in number of bytes. For the transfer of data from VSE/POWER, this buffer may be up to 65,535 bytes long.

MAPXPCCB

The macro causes a DSECT of the XPCCB to be generated.

The macro has no operands.

Format of the Macro

```
►►──┬──────┬──MAPXPCCB──────────────────────────────────────►◄
    └─name─┘
```

For name, you may assign to the DSECT a label of your own choosing. For a full description of the MAPXPCCB macro, consult the *z/VSE System Macros Reference*, SC34-2638 publication.

XPCC

The XPCC macro initiates a cross-partition communication service.

The operands, fields, and reason codes described below list a subset of operands used by VSE/POWER. For a full description of the XPCC macro, consult the *z/VSE System Macros Reference*, SC34-2638 publication.

Macro Format

```
►►─┬──────┬─XPCC XPCCB=─┬─address─────┬─,FUNC=─┬─function──┬──────────────►
   └─name─┘             ├─(1)─────────┤        └─(reg_no.)─┘
                        └─(S,address)─┘

►─┬────────────────────────────┬───────────────────────────────────────►◄
  └─,BUFFER=─┬─bflstadr──┬─────┘
             ├─(reg_no.)─┤
             └─(S,addr)──┘
```

Requirements for the caller:

AMODE:
24 or 31

RMODE:
24 or ANY

ASC Mode:
Primary

XPCCB=address|(1)|(S,address)
The operand defines the address of the XPCCB, a control block containing request-related information. For more details about the block, see "XPCCB" on page 211.

The address of the XPCCB is treated as a 3-byte address if the issuer of the macro is operating in 24-bit mode and as a 4-byte address if in 31-bit mode.

FUNC=function|(reg-no.)
The operand defines the type of request. For `function` you can specify:

CONNECT
To have the system provide a communication path to VSE/POWER. Your program can have several communication paths set up by using for every path a separate copy of the XPCCB you used for program identification (FUNC=IDENT).

DISCONN
To have the system disconnect the currently used (and no longer required) communication path to VSE/POWER.

DISCPRG
To have the system remove the existing communication path (set up by a FUNC=CONNECT) at once. This may interrupt the transfer of data from your program to VSE/POWER or vice versa.

For *GET-service* processing, VSE/POWER retains the affected queue entry with its disposition and priority unchanged.

For *PUT-service* processing, the interrupted submission has to be restarted either at a checkpoint or from the beginning.

IDENT
To make your program known to the system as a spool-access support user. This is, in effect, a "logon" service.

SENDR
To have VSE/POWER process the desired service and provide a reply.

TERMIN
To finish using the spool-access support. This is, in effect, a “logoff” service. Specify this operand if none of your program's tasks requires any further access to VSE/POWER services.

BUFFER=bflstadr|(reg-no.)|(S,addr)
The operand may be used to define your program's send buffer. If you use the operand, the system ignores the area definition given by BUFFER=specification in the associated XPCCB macro.

In the operand, `bflstadr` must point to an address list as shown below:

Bytes	Description
0	Bit 0: indicator X'00': not last entry in list X'80': last entry in list Bits 1-7: 24-bit mode: ignored 31-bit mode: bits 0-6 of address of buffer segment
1-3	24-bit mode: address of buffer segment 31-bit mode: bits 7-30 of address of buffer segment
4-7	length of buffer segment

Up to 256 entries of this format may be specified in a buffer address list. Buffer segments defined therein are concatenated for the SENDR request and are passed as one buffer to VSE/POWER.

Return Information

The system supplies return information in register 15 and in field IJBXRETC of the XPCCB; it may supply additional return information in field IJBXREAS. You should test this information along with testing the posting of IJBXSECB, the send-event control block.

VSE/POWER supplied return information in the XPCCB's user data area (field IJBXRUSR) is listed in the preceding chapters that discuss CTL, GET, PUT, and GCM service requests.

Table 77 on page 215 lists the mnemonics that you can use to test the return and reason codes supplied by the system. This list is followed by a short description of these mnemonics (Table 78 on page 216 and Table 79 on page 217). The mnemonics are also listed and described in the DSECT generated by the assembler for the MAPXPCCB macro.

Table 77. Mnemonic of Return and Reason Codes for XPCC Macro

	Mnemonic in XPCCB Field		FUNC=					
Reg.15	**IJBXRETC**	**IJBXREAS**	**CONNECT**	**DISCONN**	**DISPRG**	**IDENT**	**SENDR**	**TERMIN**
00	IJBXREOK		X	X	X	X	X	X
04	IJBXAPSP					X		
	IJBXDAPP					X		
	IJBXNIDN		X					
	IJBXNCNN		X					
08	IJBXCBSY						X	
	IJBXNDC1			X				
	IJBXNDC2			X				
	IJBXNOC1						X	
	IJBXNOC2						X	
	IJBXNOC3						X	
	IJBXNOSY		X					
	IJBXNSTO		X			X		
	IJBXNTRM							X
	IJBXQSCE		X					
	IJBXTMCR		X		X			
	IJBXWCBA			X				
	IJBXWCBK		X	X	X	X	X	X
	IJBXWIDK		X					X
	IJBXWIND						X	
	IJBXWLST						X	
	IJBXWOWN			X	X		X	X
	IJBXWPID			X	X		X	
		IJBXCPRG					X	
		IJBXDISC					X	
		IJBXABDC					X	
12	The request was rejected because the XPCCB address is invalid.							

Table 78. Return Codes (IJBXRETC) for XPCC macro

Mnemonic	Equated Value	Meaning
IJBXREOK	00	Request completed successfully.
IJBXAPSP	02	Identification is requested with the same application from the same partition. Connect to VSE/POWER is possible.
IJBXCBSY	12	The communication path to be used is busy.
IJBXDAPP	01	Identification is requested with the same application from a different partition. Connect to VSE/POWER is possible.
IJBXNCNN	05	VSE/POWER has identified itself but not issued a CONNECT request (see the Note below).
IJBXNDC1	15	The communication path is being used (a request from your program is being processed).
IJBXNDC2	16	The communication path is being used (a request issued by VSE/POWER is being processed).
IJBXNIDN	04	VSE/POWER has not yet identified itself to the system (see Note below).
IJBXNOC1	18	The communication path to be used does not exist.
IJBXNOC2	19	VSE/POWER came to a normal end of processing.
IJBXNOC3	1A	VSE/POWER came to an abnormal end of processing.
IJBXNOSY	0F	The name specified for TOAPPL in XPCCB is not SYSPWR.
IJBXNSTO	0E	No storage available for setting up the required control blocks.
IJBXNTRM	14	Your program issued FUNC=TERMIN prior disconnection.
IJBXQSCE	17	VSE/POWER is being shut down.
IJBXTMCR	0D	Too many CONNECT requests were issued by the requestor.
IJBXWCBA	1C	The request uses an XPCCB other than the one used with the FUNC=CONNECT request for setting up the communication path.
IJBXWCBK	06	The XPCCB has an invalid format.
IJBXWIDK	07	Wrong system-assigned cross-partition ID.
IJBXWIND	0A	In the defined buffer list, at least one of the indicators is wrong.
IJBXWLST	0B	One of the following: - Too many buffers are specified. - The total length of the buffers exceeds 16MB. - One of the buffers in the list has a length of zero.
IJBXWOWN	09	The task that issued the request is not authorized to use the communication path.
IJBXWPID	08	Wrong system-assigned path ID.
Note: Have your program wait for field IJBXCECB to be posted.		

Table 79. Reason Codes (IJBXREAS) for XPCC macro

Mnemonic	Equated Value	Meaning
IJBXCPRG	01	VSE/POWER issued a disconnect request as result of PSTOP SAS command (see Note 1 below). This code is logically added with IJBXDISC code (by OI instruction).
IJBXDISC	40	VSE/POWER issued a disconnect request (see Note 2 below).
IJBXABDC	80	VSE/POWER was disconnected as a result of an abnormal end (see Note 2 below).

Note:

1. Refer ro "Additional Considerations" on page 151.
2. If R15 returns X'0C', the XPCCB address is invalid. The reason code, if it occurs, is inserted into field IJBXREAS by an OI instruction.

PWRSPL

You can use the macro to do one of the following:

- generate an SPL
- update an SPL
- generate DSECTs of the SPL and of the various request control records and VSE/POWER-response records

For complex applications, the macro may not offer the scope of required control. The macro does not offer the required scope of control for applications involving the submission of output.

If the macro's scope does not meet your application's requirements, first use the macro to generate an SPL or update an existing SPL, then provide for setting up certain fields of the SPL by coding of your own. You do this by accessing the applicable SPL (field) via a generated DSECT. For the layout of the PWRSPL DSECT, please refer to "Spool-Access Support Parameter List (PWRSPL DSECT)" on page 231.

Format 1: Generating an SPL

```
►►──┬──────┬──PWRSPL TYPE=GEN──┬────────────┬──┬──────────────┬────────────────►
    └─name─┘                   └─,PRFX=xxx──┘  └─,REQ=──┬─CTL─┤
                                                        ├─GET─┤
                                                        ├─PUT─┤
                                                        └─GCM─┘

                          ┌─,CLASS=A─────┐  ┌─,JOBN=AUTONAME─┐  ┌─,PWD=──────────┐
►──┬───────────────────┬──┼──────────────┼──┼────────────────┼──┼────────────────┼──►
   └─,QUEUE=──┬─RDR─┬──┘  └─,CLASS=class─┘  └─,JOBN=jobname──┘  └─,PWD=password──┘
              ├─LST─┤
              ├─PUN─┤
              └─XMT─┘
```

```
►─┬──────────────────┬─┬─────────────────────┬──────────────────────────────────────►
  └─,USERID=user_id──┘ └─,MODE=─┬─RESET───┬──┘
                                ├─APPEND──┤
                                ├─RESTART─┤
                                ├─BROWSE──┤
                                └─GENERIC─┘

►─┬──────────────────────────────────────────┬─┬────────────────────────┬─────────────►
  │                                  (1)     │ │                   (2)  │
  └─,FUNC=─┬─(ALTER─┬─,CLASS──┬─)─┬──────────┘ └─,NEWVAL=newvalue───────┘
           │        ├─,DISP───┤   │
           │        ├─,COPY───┤   │
           │        ├─,CMPACT─┤   │
           │        ├─,NODE───┤   │
           │        ├─,REMOTE─┤   │
           │        ├─,PRI────┤   │
           │        ├─,SYSID──┤   │
           │        └─,USER───┘   │
           ├─DISPLAY──────────────┤
           ├─CANCEL───────────────┤
           ├─RELEASE──────────────┤
           ├─HOLD─────────────────┤
           ├─DELETE───────────────┤
           └─COMMAND──────────────┘

►─┬───────────────────────────────┬─────────────────────────────────────────────────►◄
  └─,OPT=─┬─RESET─────────────────┤
          │   ┌─,─────────┐       │
          │   ▼           │       │
          └─(───┬─ALLCPY──┴─)─────┘
                ├─CTLREC─┤
                ├─FORMAT─┤
                ├─NOWAIT─┤
                └─RETSEP─┘
```

Notes:

1 Valid only if REQ=CTL.

2 Valid only if FUNC=(ALTER,...)

Format 2: Updating an SPL

```
►►─┬──────┬─PWRSPL TYPE=UPD─┬─────────────┬─,SPL=─┬─splname─┬─────────────────────►
   └─name─┘                 └─,PRFX=xxx───┘       └─(reg)───┘

►─┬───────────────┬─┬─────────────────┬─┬───────────────────────┬──────────────────►
  └─,REQ=─┬─CTL─┬─┘ └─,QUEUE=─┬─RDR─┬─┘ └─,CLASS=─┬─(reg)───┬─┘
          ├─GET─┤             ├─LST─┤             └─class───┘
          ├─PUT─┤             ├─PUN─┤
          └─GCM─┘             └─XMT─┘

►─┬──────────────────────────┬─┬──────────────────────────┬─┬──────────────────────────┬──►
  └─,JOBN=─┬─(reg)─────────┬─┘ └─,JNUM=─┬─(reg)─────────┬─┘ └─,JSUF=─┬─(reg)─────────┬─┘
           └─fieldname─────┘            └─fieldname─────┘            └─fieldname─────┘
```

```
►──┬───────────────────────┬──┬──────────────────────────┬──┬──────────────────────┬──►
   └─,PWD=──┬─(reg)───────┬┘  └─,USERID=──┬─(reg)───────┬┘  └─,MODE=──┬─RESET────┬┘
            └─fieldname───┘               └─fieldname───┘             ├─APPEND───┤
                                                                      ├─RESTART──┤
                                                                      ├─BROWSE───┤
                                                                      └─GENERIC──┘

►──┬──────────────────────────────────────────┬──┬────────────────────────────┬──►
   │                                      (1) │  │                        (2) │
   └─,FUNC=──┬─(ALTER──┬─,CLASS───┬─)─┬───────┘  └─,NEWVAL=──┬─(reg)───────┬──┘
             │         ├─,DISP────┤   │                      └─fieldname───┘
             │         ├─,COPY────┤   │
             │         ├─,CMPACT──┤   │
             │         ├─,NODE────┤   │
             │         ├─,REMOTE──┤   │
             │         ├─,PRI─────┤   │
             │         ├─,SYSID───┤   │
             │         └─,USER────┘   │
             ├─DISPLAY────────────────┤
             ├─CANCEL─────────────────┤
             ├─RELEASE────────────────┤
             ├─HOLD───────────────────┤
             ├─DELETE─────────────────┤
             └─COMMAND────────────────┘

►──┬──────────────────────────────┬────────────────────────────────────────────►◄
   └─,OPT=──┬─RESET───────────────┘
            │     ┌─,──────────┐ │
            │     ▼            │ │
            └─(───┬─ALLCPY──┬──┴─)─┘
                  ├─CTLREC──┤
                  ├─FORMAT──┤
                  ├─NOWAIT──┤
                  └─RETSEP──┘
```

Notes:

1 Valid only if REQ=CTL.

2 Valid only if FUNC=(ALTER,...)

Format 3: Generating a DSECT

```
►►──┬────────┬──PWRSPL TYPE=MAP──┬──────────────┬──────────────────────────►◄
    └─name───┘                   └─,PRFX=xxx────┘
```

Since many operands in the various formats are identical, the operands are explained only once.

TYPE=GEN|MAP|UPD
The operand specifies the desired type of macro expansion:

GEN
Causes an SPL to be generated.

MAP
Causes a DSECT of the SPL to be generated. Only the operand PRFX=xxx is meaningful together with a TYPE=MAP specification.

UPD
Requests that an SPL defined in the program by PWRSPL TYPE=GEN be updated in accordance with the specified operands. SPL fields corresponding to omitted operands remain unchanged.

CLASS=class|(reg)
The operand specifies the class that:

- Matches the class of the queue entry which is to be retrieved (REQ=GET is specified).
- Is to be assigned to the output queue entry (REQ=PUT is specified).
- Is to be used as a search argument if a control function is to be processed (REQ=CTL is specified).

As class value, specify the desired one-byte input or output class as self-defining character constant.

If the macro specifies TYPE=GEN, then:

- You cannot use register notation.
- Class A is used as default.

If you use a register (together with TYPE=UPD), it must point to a one-byte field that contains the class value.

FUNC=(ALTER,attrib-type)|CANCEL|COMMAND|DELETE|DISPLAY| HOLD|RELEASE
The operand applies only if you specify also REQ=CTL; it specifies the type of function to be performed.

ALTER
Causes VSE/POWER to alter, in the specified queue for the named job(s) (with optional jobnumber and/or jobsuffix and/or class (see note), the attribute that you specify for attrib-type.

For *attrib-type*, you can specify one of the attributes discussed under NEWVAL=field|(reg), below. Only one attribute can be changed per CTL request.

Note: Although class is optional, it is automatically provided as default 'A' through PWRSPL TYPE=GEN. Consider overwriting the default and specify another class value using the CLASS=class operand. You can also nullify the default class value (and address queue entries in more than one class) using the PWRSPL TYPE=UPD,CLASS=(reg) request, where (reg) points to a blank (X'40') field. Alternatively, you can nullify the default class value by setting the SPLGCL field to X'40' directly in the SPL control block before you send the SPL to VSE/POWER.

CANCEL
Causes VSE/POWER to cancel (flush) the job identified by job name and, optionally, by job number.

COMMAND
Indicates that VSE/POWER is to process the command supplied in the field SPLCFLD of the SPL that is being generated or updated. VSE/POWER accepts the command without error checking for the command.

For passing a command to VSE/POWER, the following rules have to be observed:

- The command must be set up using uppercase letters.
- The command cannot be longer than 130 bytes.

- Continuation of the command is not supported.
- At least one blank must follow the command within the 130 byte area.
- For a coding example, see label 'CTLAB1' of the PWRSASEX example in Chapter 13, “Spool-Access Support Programming Example,” on page 271.

Here, successful processing of the command is not indicated by a message. For details, see “Retrieving Messages” on page 67.

The table below lists the commands that VSE/POWER accepts via a spool-access communication path:

```
PALTER queue entries   (see Note 1 below)
PBRDCST
PCANCEL jobname        (see Note 1 below)
PDELETE queue entries  (see Note 1 below)
PDELETE FCB            (see Note 2 below)
PDELETE MSG
PDISPLAY queue entries (see Note 6 on page 222 below)
PDISPLAY CRE
PDISPLAY DEL
PDISPLAY TOTAL   PDISPLAY M   PDISPLAY MSG
PDISPLAY A
PDISPLAY BIGGEST
PDISPLAY Q   PDISPLAY T
PDISPLAY TASKS
PDISPLAY DYNC
PDISPLAY PNET
PDISPLAY EXIT
PDISPLAY SPDEV
PDISPLAY SPDEVT
PDISPLAY STATUS   PDISPLAY AUSTMT
PDISPLAY TAPE          (see Note 2 below)
PDISPLAY VIO           (see Note 2 below)
PFLUSH DEV|cuu         (see Notes 2 and 3 below)
PGO DEV|cuu            (see Notes 2 and 3 below)
PHOLD queue entries    (see Note 1 below)
PINQUIRE
PLOAD DYNC             (see Note 2 below)
PRELEASE queue entries (see Note 1 below)
PRESTART DEV           (see Note 2 below)
PSEGMENT               (see Note 5 on page 222 below)
PSETUP DEV             (see Note 2 below)
PSTART DEV|cuu         (see Notes 2 and 3 below)
PSTOP DEV|cuu          (see Notes 2 and 3 below)
PVARY DYNC             (see Note 2 below)
PVARY MSG              (see Note 2 below)   PXMIT node-id
PXMIT DEV              (see Note 2 below)
```

Note:

1. Accepted only if there is a match of the recorded and specified user IDs (origin or target) and, if applicable, also of these passwords.

   Only the owner of an entry can manipulate a target entry with a destination of ANY (unless his own user ID is ANY which, however, is not recommended).
2. Only for authorized users – for example the subsystem administrator, if there is one.
3. Messages which are due to a syntax error of the command are routed to the application program issuing the command. For example:

   ```
   1R52I PSTART OPERAND 1 MISSING OR INVALID
   or
   1R58I PSTART DEVICE 00E IS IN USE
   ```

If the command has been processed successfully, the application program gets return/feedback code (00/01). If within the command 'cuu' has been specified, messages, which are related to this command, may be issued and routed to the console, but not to the application program. For example:

```
1Q34I RDR WAITING FOR WORK ON 00C
or
1Q33I STOPPED LST, 00E
```

4. For a summary of command access limitation, see "Scope of GET/CTL Access to Queue Entries" on page 61.
5. The command addresses a queue entry in creation and is accepted only if there is a match of the recorded and specified IDs (origin or target) and, if applicable, also of these passwords.
6. You can display queue entries of the physical RDR/LST/PUN/XMT queues and of the logical CRE and DEL queues.

DELETE
Causes VSE/POWER to delete, in the specified queue, the named job(s) (further qualified by optional jobnumber and/or jobsuffix and/or class (see note for ALTER)).

DISPLAY
Causes VSE/POWER to return information about the queue entries of the specified queue as described by jobname (further qualified by optional jobnumber and/or class (see note for ALTER)).

HOLD
Causes VSE/POWER to change, in the specified queue, the disposition of the named job(s) (further qualified by the optional jobnumber and/or jobsuffix and/or class (see note for ALTER)) to the following:

H (hold) if it was D (dispatchable)

L (leave) if it was K (keep).

RELEASE
Causes VSE/POWER, in the specified queue for the named job(s) (further qualified by optional jobnumber and/or jobsuffix and/or class (see note for ALTER)), to take them out of the hold or leave state and make them available for processing.

JOBN=jobname|fieldname|(reg)
The operand specifies the VSE/POWER job name that is to be used for the execution of the request. The job name you specify must be alphameric (more precisely: "alphaj" as defined in Chapter 14, "Return and Feedback Codes and Their Meanings," on page 297 and, for CTL requests, those of generic type as well) and not longer than eight characters.

If the macro specifies TYPE=GEN, then:

- You must define the name as a self-defining character constant.
- Omission of the operand causes AUTONAME to be used as the default name.

If you code this operand together with TYPE=UPD, specify for jobname the label of an eight-byte field that contains the job name left justified and padded with blanks.

If you use a register, it must point to an eight-byte field with the name.

JNUM=fieldname|(reg)
The operand can be used only together with TYPE=UPD. It specifies the

number which VSE/POWER assigned to the job whose queue entry is to be manipulated or whose data is to be retrieved.

If you use a register, it must contain the job number. If you do not use a register, the field name must be the label of a halfword that contains the job number in binary notation.

If you do not want to supply a job number, set the field (or register) to binary zeros.

JSUF=fieldname|(reg)

The operand can be used only together with TYPE=UPD. It specifies the job-suffix (segment) number assigned to the queue entry that is to be manipulated or to be retrieved.

If a register is used, it must contain the number. If you do not use a register, the field name must be the label of a halfword containing the number (in binary).

MODE=APPEND|BROWSE|GENERIC|RESET|RESTART

The operand specifies the mode of operation for the requested service:

APPEND

Spooling is to continue at the end of an already existing queue entry. This applies only to the PUT output function.

BROWSE

Useful primarily if you intend to examine (but not to update) a job.

If you specify BROWSE, you must also provide the name of the job to be accessed, with or without the applicable job number and job suffix. A queue entry accessed in BROWSE mode enters the active (DISP=*) state. Because viewing can only be terminated by the QUIT request of the GET service, the queue entry is left unchanged in its queue even if the entry's disposition is D. For more details on browsing, refer to “Browsing a Queue Entry for Viewing Only” on page 78.

For a retrieval in BROWSE mode, VSE/POWER accepts only a subset of GET-service requests (with an action code in byte PXUACT1 of the XPCCB field IJBXSUSR) as shown below:

Type of Request	Mnemonic Equated to the Action Code
A retrieval request	PXUATSDR
A quit request	PXUATABR
A restart request	Not applicable. You submit this request by passing (to VSE/POWER) a restart-control record.

GENERIC

Causes VSE/POWER to retrieve the first eligible queue entry destined for a certain user within the specified class. When processing a retrieval request in this mode, VSE/POWER ignores the specification of a job name, a job number, or a job suffix.

VSE/POWER selects, for retrieval, the queue entry whose characteristics are closest to the ones defined in the PWRSPL macro.

You can include in your request up to three additional classes by:

1. Inserting the additional classes left justified in the field SPLGNV of your SPL followed by a blank.
2. Setting the flag SPLGOACL in byte SPLGOPT, the SPL's option byte.

You can further limit the selection of retrieved queue entries to those whose target (disregarding 'from') user ID matches the user ID of the GET request (SPLGUS) by setting the flag SPLGO2HU in byte SPLGOPT2, the option byte 2 of the SPL.

RESET
Causes the mode settings to be reset to the default values.

RESTART
Spooling is to begin at a certain record of an already existing queue entry (PUT-output function). This record is either of the following:

- The one whose number your program supplies in field SPLDCREC of the applicable SPL.
- The record last checkpointed by VSE/POWER if your program does not supply a record number in this SPL field.

NEWVAL=newvalue|fieldname|(reg)
The operand names the direct constant or field that contains the new value to be used by VSE/POWER as attribute for the named queue entry.

If you have specified TYPE=GEN in this macro before, the *new value* must in all cases be defined as a self-defining character constant.

If you have specified the FUNC= operand in this macro before with (ALTER,attrib-type), this operand gives you the new value to be used. What type of value it is, is explained below. These are the parameters you have to choose from for the FUNC= operand as 'attrib-type':

CLASS
The name of a one-byte field that contains the new class of the queue entry. If you use a register, it must point to the one-byte field.

DISP
The name of a one-byte field that contains, in character format, the new disposition (D, K, H, or L) of the affected queue entry. If you use a register, it must point to the one-byte field.

For further information on disposition refer to the *VSE/POWER Administration and Operation*, SC34-2625 publication.

How the operator is to handle dispositions X and Y is described in the Chapter “Operating with VSE/POWER” of that publication.

COPY
The name of a three-byte field that contains, in character format, the new number of copies (any value from 1 to 255). If you supply the number right justified, leading zeros are required.

If a register is used, it must point to the three-byte field.

The specification applies only to output queue entries; it is ignored if you specify it for an input queue entry.

CMPACT
The label of a four-byte field which contains the name of the new compaction table set to be used for transmitting the queue entry to an SNA workstation. Supply this table set's name left justified without leading blanks.

Instead of the name of a compaction table, you may specify either:

`*` To indicate that the default compaction table is to be used.

NO To indicate that no compaction should be performed.

If you use a register, it must point to the four-byte field.

NODE
The label of an eight-byte field that contains, in character format, the new target destination of the queue entry. In this field, supply the destination left justified without leading blanks. If you use a register, it must point to the eight-byte field.

REMOTE
The label of a three-byte field that contains, in character format, the new remote ID. This is a value from 0 to 250.

If you supply the number right justified, leading zeros are required. If you use a register, it must point to the three-byte field.

The specification applies only to output queue entries; it is ignored if you specify it for an input queue entry.

PRI
The name of a one-byte field that contains, in character format, the new priority. If you use a register, it must point to the one-byte field.

SYSID
The label of a one-byte field that contains, in character format, the new system ID. If you use a register, it must point to the one-byte field.

USER
The label of an eight-byte field that contains, in character format, the new target user ID of the queue entry. In that field, supply this ID left justified without any leading blanks. If you use a register, it must point to the eight-byte field.

OPT=RESET|(service-options)
The operand specifies options for performing the requested service.

RESET
Causes VSE/POWER to reset (turn off) any option specified previously.

(service-options)
You may omit the enclosing parentheses if you specify only one of the options.

If a specified option does not apply to the requested function, VSE/POWER ignores this option.

ALLCPY
Causes VSE/POWER to return all copies of an output queue entry to the requestor. The specification applies only if you specified REQ=GET.

Depending on OPT=CTLREC, each copy ends with its last data or control record. When OPT=CTLREC then a control record follows with a record prefix and one blank byte of data. The command code is X'07', the record type is X'08', meaning 'end-of-copy', and the record number is zero.

In all cases the new copy starts with an inline SPL record of command code X'00', record type X'01' and a record number of zero. Depending on OPT=CTLREC, either the first control record or the first data record of the new copy then follows, starting with record number one or more.

CTLREC
Causes VSE/POWER to return also immediate control records (such as skip to channel 1 and space 2 lines) when retrieving an output queue entry.

A control record consists of a record prefix and one byte of data. The command code is contained in the record prefix. For control records with a command code reserved for use by VSE/POWER only, see "Spooling of Records with Carriage Control Character X'FE'" on page 120.

FORMAT

Causes the result of a requested RDR/LST/PUN/XMT or CRE/DEL/TOTAL queue display to be returned as 'fixed' format records rather than console-display (also called 'free') format messages.

For a PDISPLAY BIGGEST request, the FORMAT option is ignored.

The specification applies to queue display commands, set up either as fixed format commands by

```
PWRSPL REQ=CTL,FUNC=DISPLAY,QUEUE=...
```

or as free format commands by

```
PWRSPL REQ=CTL,FUNC=COMMAND
```

The terms 'fixed' and 'free' format command and 'fixed'/'free' format messages have no relation to each other. For use of both command formats, refer to labels CTLA1 and CTLAB1 in Chapter 13, "Spool-Access Support Programming Example," on page 271.

For the receipt of 'fixed' format messages, keep the following in mind:

- They are structured according to the PXFMDSCT DSECT. For the layout of the Fixed Format Queue Display Record, refer to "Spool-Access Support Parameter List (PWRSPL DSECT)" on page 231.
- The standard 8-byte prefix (see RECPRFIX in "Spool-Access Support Parameter List (PWRSPL DSECT)" on page 231) identifies fixed format records by RECTYPE=RECTFIXM.
- Only actual queue record information is passed to your program, i.e., headline messages for the various queues are suppressed.
- The queue type of a presented queue entry can be derived from field PXFMQUID. It contains R|L|P for RDR/LST/PUN data types, whereas the additional flag PXFMFLG1.PXFMF1XQ shows that the entry actually resides in the XMT queue.
- The "being browsed" informaton is included in fields 'PXFMMACN, PXFMMAC1,...,PXFMMAC9'. For a non-shared VSE/POWER system, 'PXFMMACN', showing a nonzero value is equivalent to the '*' in the 'B' column of a normal queue display. If for a shared system at least one field of 'PXFMMAC1,...,PXFMMAC9' is nonzero, this is also equivalent to '*' in the 'B' column.
- The queue type of a presented queue entry of the DELETION QUEUE shows the original queue type before the entry was deleted. PXFMFLG3.PXFM3DEL shows that the entry is in deletion.
- The queue type of a presented queue entry of the CREATE QUEUE shows the desired queue type in field PXFMQUID. PXFMFLG3.PXFM3CRE shows that the entry is in creation.

  The following new fields defined in PXFMDSCT are meaningful only if the fixed-format messages are returned for a 'PDISPLAY CRE' request.

  – PXFMTASK contains the owning task identifier (last 7 bytes) as shown in 'PDISPLAY TASKS'.
  – PXFMOWNT contains the task type (1 byte):
    - 'J' - output is being created by a JOB

  - 'N' - job or output is being received from other NODE
  - 'R' - job is being received from REMOTE station
  - 'S' - job or output is being spooled by SAS application
  - ' ' - job or output is being created by other task type (none of the above)
- PXFMOWND contains the owner description (8 bytes) in relation to the task type shown in PXFMOWNT.
  - jobname
  - node name
  - Remote ID
  - Application name
  - blank

For a LST/PUN queue entry the start time (PXFMSTRT) and stop time (PXFMSTOP) are identical until the queue entry is printed/punched and requeued.

For detailed scheduling information as offered by PXFMFLG2, refer to *VSE/POWER Administration and Operation*, SC34-2625.

NOWAIT

Requests control to be returned to the requestor when a wait condition occurs because of lack of disk space. If you do not specify NOWAIT, VSE/POWER waits for such space to become available. What this means for your program is discussed below.

During *GET-service processing*, VSE/POWER may find that the account-file is full. The situation may come up when VSE/POWER executes one of the following subfunctions:

- CLOSE the processing for the currently accessed queue entry.
- Perform a FLUSH-HOLD for the spool request (applies only to an application involving external device support).
- Purge the involved queue entry.
- Quit processing the request.

When the account-file-full condition occurs, the function has already been performed. Therefore, VSE/POWER cannot inform your program right away. If no other GET (CTL or PUT) request follows, your program does not become aware of this situation. However, a subsequent GET (CTL or PUT) request from your program is rejected with applicable return and feedback codes supplied by VSE/POWER. In your program, you can then decide, whether you want to wait and retry after a certain time interval or to set up a new communication path to VSE/POWER to start the new function.

During a *PUT-service processing*, VSE/POWER may run into a “short of disk space” situation as indicated:

- While VSE/POWER is spooling a job or job output.

  VSE/POWER stops further spooling of the submitted job or output. This results in the following:

  - If an output file is being spooled without being checkpointed, this file is lost, and VSE/POWER queues an information message. If the file is checkpointed, VSE/POWER queues the file up to last committed checkpoint; the file's remaining data is lost.

- If a single job is being spooled, the job is lost, and VSE/POWER queues an information message.
- If multiple jobs are being spooled, the jobs already spooled are kept in the input queue, but the job being processed and all subsequent ones are lost. VSE/POWER queues a message and returns a verification SPL for the last job spooled successfully.
- If a segment-output request is being processed, VSE/POWER may or may not pass to your program a verification SPL in addition to the data-file-full indication.

  By passing this SPL, VSE/POWER informs your program that all data submitted up to this point has been spooled and a queue entry for the segment exists. Processing for building another segment cannot continue.

  If VSE/POWER passes only the data-file-full indication, all data submitted since the last successful segment request or checkpoint (whichever applies) is lost.

• When VSE/POWER tries to write into the account file.

  This can occur after a CLOSE, quit, or segment request; it can occur after a spool-data request during multiple-job submission.

  - For a CLOSE or quit request, VSE/POWER returns a successful completion indication. However, VSE/POWER rejects any subsequent PUT, GET or CTL function as long as the account-file-full situation exists. For the new function request, VSE/POWER performs no error checking; instead it returns to your program the return- and feedback-code combination PXPRCOKF and PXPS04SOA to indicate that the account file is full.
  - For a segment request, VSE/POWER returns the PXPRCOKF/ PXPS0SOA return/feedback-code combination together with the SPL that describes the output segment just queued. VSE/POWER does not accept any further spool requests.
  - For multiple-job submission (with one open PUT-service request), VSE/POWER returns a verification SPL together with the PXPRCOKF/PXPS04SOA return/feedback code combination. This SPL applies to the job that was queued, but for which no account record could be written. Any subsequent job-spool requests are rejected by VSE/POWER.

RETSEP

Causes separator pages (or cards) to be returned as normal data records in front and at the end of the requested output queue entry. Separator pages (cards) that VSE/POWER builds are passed with their MCCs.

The option applies only if you specified REQ=GET and if a JSEP value other than zero was specified for the queue entry.

PRFX=xxx

Use this operand if, for the generated SPL or SPL DSECT, you want the field names to begin with characters other than SPL. This avoids the occurrence of duplicate names if your program includes the macro two or more times; for example several times with TYPE=GEN and once with TYPE=MAP.

For xxx, specify the string of up to three characters with which you want the field names to begin.

PWD=password|fieldname|(reg)

The operand specifies the password associated with the queue entry to be retrieved or manipulated. The password must be alphameric and not longer than eight characters; if it is shorter, it is to be defined left justified and padded with blanks.

For a request with TYPE=GEN, specify a password (if this is feasible) as a self-defining character constant.

For a request with TYPE=UPD, specify the label of an eight-byte field that contains the password. If you use a register, it must point to the eight-byte field that contains the password.

For a request with TYPE=GEN, VSE/POWER defaults to a password of eight blanks. Then you may access all jobs without a password -- either read-in locally or submitted by a spool-access PUT request.

QUEUE=LST|PUN|RDR|XMT

The operand specifies the queue that is to be accessed. The queue specifications valid for the various function requests are indicated by an X in the table below:

	QUEUE=			
Function Specification	**LST**	**PUN**	**RDR**	**XMT**
REQ=CTL	X	X	X	X
REQ=GET	X	X	X	X[1]
REQ=PUT: spooling job(s)	-	-	X	-[2]
REQ=PUT: spooling output	X	X	-	-[3]

[1] Applicable only for "Direct Queue Entry Access"
[2] If your program submits a job for processing at another node, it must define this in the * $$ JOB statement for the job.
[3] If your program submits output data for transmission to another node, the target node's name and user ID must be defined in the fields SPLDTNN and SPLDTUID, respectively, of the applicable SPL.

REQ=CTL|GET|PUT|GCM

The operand specifies the type of function to be performed. The set of operands that applies to each of these basic function requests is given below. In the operand lists, M = mandatory and O = optional. Specify:

CTL

To pass to VSE/POWER a control request or a command for execution. Operands that apply to a CTL request (where: M = mandatory; O = optional):

```
FUNC=function                          M
JOBN=jobname|(reg)                     M1
QUEUE=RDR|LST|PUN|XMT                  M1
USERID=user-id|(reg)                   M

CLASS=class|(reg)                      O2
JNUM=fieldname|(reg)                   O
JSUF=fieldname|(reg)                   O
NEWVAL=field|(reg)                     O
OPT=FORMAT                             O
PWD=password|(reg)                     O
```

[1]Optional if your program passes a command to VSE/POWER specified directly in the field 'SPLCFLD' and FUNCTION is COMMAND.

[2]If omitted, then VSE/POWER uses the default class A.

GET

To retrieve, from the specified VSE/POWER queue, the named queue entry. Operands that apply to a GET request (where: M = mandatory; O = optional):

```
CLASS=class|(reg)                         M
JOBN=jobname|(reg)                        M
QUEUE=RDR|LST|PUN|XMT                     M¹
USERID=user-id|(reg)                      M

JNUM=fieldname|(reg)                      O
JSUF=fieldname|(reg)                      O
MODE=BROWSE                               O
MODE=GENERIC                              O
OPT=(ALLCPY,CTLREC,NOWAIT,RETSEP)         O
PWD=password|(reg)                        O

¹XMT is applicable only for "Direct Queue Entry Access"
```

PUT

To have job(s) spooled to VSE/POWER input queues (RDR, XMT) and Job output to the VSE/POWER output queues (LST, PUN, XMT).

Operands that apply to a PUT-job request (where: M = mandatory; O = optional):

```
QUEUE=RDR                                 M
USERID=user-id|(reg)                      M

OPT=NOWAIT                                O
PWD=password|(reg)                        O
```

Operands that apply to a PUT-output request (where: M = mandatory; O = optional):

```
QUEUE=LST|PUN                             M
JOBN=jobname|(reg)                        M
USERID=user-id|(reg)                      M

CLASS=class|(reg)                         O¹
MODE=APPEND|RESTART                       O
OPT=NOWAIT                                O
PWD=password|(reg)                        O

¹If omitted, then  VSE/POWER uses the default class A.
```

Spooling of output data may require that your program set up a certain number of SPL fields individually. For more information about setting up SPL fields, see "Submitting Output Data" on page 117.

Note: For spooling job(s) or output to the XMT queue, see the description of the QUEUE operand.

GCM

To have messages retrieved from a VSE/POWER fixed format job event and output generation messages queue. Operands that apply to the GCM Open-request (where M=mandatory; O=optional) are:

```
USERID=user-id|(reg)                      M
JOBN=jobname|(reg)                        O
JNUM=fieldname|(reg)                      O
```

Retrieval of fixed format job event and output generation messages requires that your program sets up some SPL fields individually. For more information, see “How to Retrieve Job Event and Output Generation Messages” on page 141.

SPL=splname|(reg)
The operand specifies the address of the SPL to be used. It applies only if you specify TYPE=UPD.

If you do not use a register, the code generated for your PWRSPL macro causes a pointer to the SPL to be loaded into register 1. Save this register's content before you issue the macro. If you code the macro with a name in the name field, that name must be identical with the symbolic address you specify for splname.

USERID=user-id|fieldname|(reg)
The operand specifies the user ID associated with the queue entry that is to be retrieved, submitted, or manipulated.

For a request with TYPE=GEN, specify the actual ID as a self-defining character constant.

For a request with TYPE=UPD, specify the label of an eight-byte field that contains the ID left justified and padded with blanks.

If you use a register, it must point to the eight-byte field that contains the ID.

Note: ANY is not recommended as user ID.
For a summary of access limitations, see “Scope of GET/CTL Access to Queue Entries” on page 61.

Spool-Access Support Parameter List (PWRSPL DSECT)

If you use the PWRSPL macro and specify TYPE=MAP, you will be provided with the following DSECTs:

Bytes Hex	Field Label	Description/Function
General Section Part 1		
00	SPLDS	Start of parameter list
00-02	SPLGHD	Storage descriptor
03	SPLGVM	Version and modification level
	SPLGVM1	X'10' - Version and modification level 10
	SPLGVM2	X'20' - Version and modification level 20
	SPLGVM3	X'30' - Version and modification level 30
	SPLGVM31	X'31' - Version and modification level 31
04-0B	SPLGJB	Job name, left justified and padded with blanks
0C-0D	SPLGJN	Job number, binary
0E	SPLGJS	Job suffix - X'00' to X'7F' (0 to 127)
	SPLGJSLA	X'80' - Last segment indication in bit 0 Actual segment number 1 - 127 in bit 1 - 7
0F	SPLGCL	Job class
10-17	SPLGPW	Job password
18-1F	SPLGUS	User id of requestor

Bytes Hex	Field Label	Description/Function
20	SPLGQI	Queue identifier
	SPLGQIR	C'R' - RDR queue
	SPLGQIL	C'L' - LST queue
	SPLGQIP	C'P' - PUN queue
	SPLGQIX	C'X' - XMT queue
21	SPLGFLG	Flag byte (for Reply SPL)
	SPLGFXR	C'R' – RDR type in XMT queue
	SPLGFXL	C'L' – LST type in XMT queue
	SPLGFXP	C'P' – PUN type in XMT queue
The following fields define the request types. The contents of the subrequest byte and function bytes depend on the request type.		
22	SPLGRQB	Request byte
	SPLGRPUT	X'01' - PUT OPEN request
	SPLGRGET	X'02' - GET OPEN request
	SPLGRCTL	X'03' - CTL OPEN request
	SPLGRGCM	X'04' - GCM OPEN request
23	SPLGSRB	Subrequest byte
	SPLGSRDY	X'01' - Display job/output queue entry
	SPLGSRCN	X'02' - Cancel job
	SPLGSRRL	X'03' - Release job/output queue entry
	SPLGSRHD	X'04' - Hold job/output queue entry
	SPLGSRDL	X'05' - Delete job/output queue entry
	SPLGSRAL	X'06' - Alter job/output queue entry
	SPLGSRCM	X'07' - VSE/POWER command
	SPLGSRDC	X'08' - Delete checkpoint information
	SPLGSRJG	X'09' - GCM: RETRIEVE JGM
	SPLGSRJC	X'0A' - GCM: RETRIEVE JCM
	SPLGSROG	X'0B' - GCM: RETRIEVE OGM
24	SPLGFB1	Function byte 1
	SPLGF1AP	X'01' - Append of incomplete queue entry
	SPLGF1RS	X'02' - Restart of queue entry
	SPLGF1BR	X'03' - Browsing of queue entry
	SPLGF1GG	X'04' - Generic GET request
	SPLGF1QM	X'05' - PUT: Queue completion message CTL-RELEASE: Queue completion message (must be specified together with SPLGFB2.SPLGF2MR flag)
	SPLGF1KM	X'06' - GCM: Retrieve and keep message
	SPLGF1DM	X'07' - GCM: Retrieve and delete message
	SPLGF1RM	X'08' - GCM: Remove message
	SPLGF1QQ	X'09' - PUT: Queue job event message
	SPLGF1PM	X'0A' - GCM: Purge message queue

Bytes Hex	Field Label	Description/Function
	SPLGF1QP	X'0B' - PUT: Queue job completion and output generation messages
	SPLGF1QO	X'0C' - PUT: Queue output generation message
	SPLGF1QX	X'0D' - PUT: Queue job event and output generation messages
	SPLGF1XS	X'0E' - GCM: Start XEM service
	SPLGF1XM	X'0F' - GCM: Retrieve and delete XEM
	SPLGF1XT	X'10' - GCM: Stop XEM service
25	SPLGFB2	Function byte 2
	SPLGF2CL	X'01' - Alter class
	SPLGF2DP	X'02' - Alter disposition
	SPLGF2CP	X'03' - Alter copy number
	SPLGF2CM	X'04' - Alter compaction table name
	SPLGF2RE	X'05' - Alter remote id
	SPLGF2PR	X'06' - Alter priority
	SPLGF2SY	X'07' - Alter system identifier
	SPLGF2TN	X'08' - Alter destination node name
	SPLGF2TU	X'09' - Alter destination user id
	SPLGF2MR	X'0A' - Release command gets completion message (must be specified together with SPLGFB1.SPLGF1QM flag)
26-2D	SPLGNV	Field containing the new value for the alter or additional classes
26-28	SPLGACLS	Extra classes for generic GET
2E	SPLGOPT	Option byte 1
	SPLGOSEP	X'80' - Return separator pages/cards
	SPLGOFCC	X'40' - Feed back immediate commands
	SPLGOALL	X'20' - Pass all copies of queue entry
	SPLGOFIX	X'10' - Return fixed format queue display
	SPLGONOW	X'08' - NOWAIT option
	SPLGOACL	X'04' - Upto 3 extra classes specified
	SPLGOGIC	X'02' - Request GET-OPEN for queue entry in creation
2F	SPLGOPT2	Option byte 2
	SPLGO2AC	X'80' - Convert ASA characters to machine control characters
	SPLGO2HU	X'40' - Honor user id for generic GET request
	SPLGO2BT	X'20' - Ignore blank truncation during spooling
	SPLGO2QN	X'10' - Use queue record number
	SPLGO2FE	X'08' - Allow to put X'FE' records
	SPLGO2OJ	X'04' - PUT: Pass original job number in job event and output generation messages - GCM: Use original job number in JCMDS/JGMDS/OGMDS to search for messages
	SPLGO2CD	X'02' - GCM: Use generated job ids
	SPLGO2WP	X'01' - GCM: GCM WAIT is allowed when in PEND state
	SPLGSLEN	Length of general section part 1

Bytes Hex	Field Label	Description/Function
General Section Part 2 The following fields contain descriptive information about the queue entry either built or accessed.		
30	SPLDDP	Disposition of queue entry
31	SPLDPR	Priority of queue entry
32-33	SPLDOJ#	Original job number
34	SPLDSID	System identifier
35	SPLDMOP	General option byte 1
	SPLDMNCS	X'20' - No copy separators
	SPLDMOHP	X'10' - Hold when printing/punching failed
36	SPLDFLG	Flag byte
	SPLDFVSE	X'80' - Output produced on z/VSE system
	SPLDFCKI	X'40' - Extended checkpoint information exists
	SPLDFCKE	X'20' - Extended checkpoint information unavailable due to I/O error
	SPLDSKIP	X'10' - SET SKIP=YES in autostart
	SPLDFRUN	X'08' – NORUN=IGN specified on * $$ JOB statement
37	SPLDCCPY	Checkpoint copy number
38-3B	SPLDRCT	Total record count
3C-3F	SPLDPCT	Total page count for list entries only
40-43	SPLDLCT	Card/line count for LST/PUN entries only
44-47	SPLDCREC	Checkpoint record number or PUT-OPEN-RESTART record number
48-57	SPLDUI	User information
58-5F	SPLDONN	Originator node name
60-67	SPLDOUID	Originator user/remote identifier
68-6F	SPLDTNN	Target node name
70-77	SPLDTUID	Target user/remote identifier
78-8B	SPLDPRGN	Programmer name
8C-93	SPLDROOM	Room number
94-9B	SPLDDEPT	Department number
9C-A3	SPLDBLDG	Building number
A4-A5	SPLDLREC	Maximum record length
• **Output Section** The following fields are only applicable when either spooling or retrieving output to/from the LST, PUN, or XMT queues.		
A6	SPLORCFM	Record format
	SPLORSCS	X'80' - SCS print
	SPLORBMS	X'40' - BMS mapping
	SPLOR327	X'20' - 3270 format
	SPLORAPA	X'10' - CPDS data stream, APA data
	SPLORESC	X'08' - Escape mode

Bytes Hex	Field Label	Description/Function
	SPLORASA	X'04' - ASA control characters
	SPLORMCC	X'02' - Machine control characters
A7	SPLONCPY	Number of copies
A8-AB	SPLOCOMP	Compaction table name (RJE,SNA output only)
AC-B3	SPLOFORM	Forms identifier (FNO)
B4-BB	SPLOEWTR	Subsystem name
BC-C3	SPLOFCB	FCB name
C4-CB	SPLOUCB	UCB name
CC	SPLOUCBO	UCB option byte
	SPLOUCBD	X'80' - Block data check option
	SPLOUCBF	X'40' - Fold option
CD	SPLONSEP	Number of separator pages/cards
CE	SPLOTDP	Output transmission disposition
CF		Reserved
• **3800 Section** The following fields are only applicable for 3800 output		
D0-DF	SPL3TAB	Character arrangement tables
D0-D3	SPL3TAB1	Character arrangement table 1
D4-D7	SPL3TAB2	Character arrangement table 2
D8-DB	SPL3T3B3	Character arrangement table 3
DC-DF	SPL3TAB4	Character arrangement table 4
E0-E3	SPL3MODF	Copy modification name
E4-E7	SPL3CCHR	Character arrangement table for copy modification
E8-EF	SPL3CPYG	Copy groupings
E8	SPL3CPG1	Copy group 1
E9	SPL3CPG2	Copy group 2
EA	SPL3CPG3	Copy group 3
EB	SPL3CPG4	Copy group 4
EC	SPL3CPG5	Copy group 5
ED	SPL3CPG6	Copy group 6
EE	SPL3CPG7	Copy group 7
EF	SPL3CPG8	Copy group 8
F0-F3	SPL3FLSH	Flash identifier
F4	SPL3FLCT	X'FF' - Flash count = 255
F5	SPL3FLG1	Flag byte 1
	SPL3F1BR	X'80' - Burst is requested
	SPL3F1TR	X'40' - 1st byte contains TRC character
	SPL3F138	X'20' - 3800 section present
	SPL3SLEN	Length of all sections to date
F6-F7		Reserved for future use

Bytes Hex	Field Label	Description/Function
• **Extended Section for SPL Version 2**		
F8-F9	SPLEOPOF	Offset to start of OPTBs
FA-FB	SPLEOPLN	Length of specified OPTBs
• **Extended Section for SPL Version 3**		
FC-103	SPLXDIST	Distribution code
104-105	SPLXQRJ#	Job number of job that created the output
106-107	SPLXCKIL	Length of checkpoint with extended information
108-10B	SPLXQNUM	Queue entry number, binary
10C	SPLXFLG1	Extended flag byte 1
	SPLX1GLN	X'80' - LOG=NO specified
	SPLX1EMG	X'40' - EOJMSG=YES specified
	SPLX1ACE	X'20' - entry created by PACCOUNT PUN
	SPLX1SNO	X'10' - PUT: output not to be spool access protected (SECAC=NO) - GET: entry not spool access protected (SECAC=NO)
	SPLX1DSP	X'08' - Direct GET for $SPLnnnn
	SPLX1XRD	X'04' - GCM-XEM: queue RDR entry event
	SPLX1XLS	X'02' - GCM-XEM: queue LST entry event
	SPLX1XPN	X'01' - GCM-XEM: queue PUN entry event
10D	SPLXOB1	Extended option byte 1
	SPLXO1CQ	X'01' - PUT: Job event and output generation messages to common queue
	SPLXO1DQ	X'02' - PUT: Job event and output generation messages to common and user queue
10E-10F	SPLXWAIT	GCM: Wait time (0 - 27962 seconds)
	SPLXWETR	X'FFFF' - GCM: Wait indefinitely
110-117	SPLXSID	z/VSE security user id
118-11F	SPLXSPW	z/VSE security password

Bytes Hex	Field Label	Description/Function
120-127	SPLXPMDE	Processing mode (PRMODE)
128-12F	SPLXPRIV	PUT: Private information
130-133	SPLXQDAT	GET: Creation date of entry (DDMMYYYY or MMDDYYYY), packed format
134-137	SPLXQTIM	GET: Creation time of entry (0HHMMSSF), packed format
138-13A	SPLXEXPM	PUT: Expiration moment values:
138-139	SPLXEXPD	PUT: Expiration days
13A	SPLXEXPH	PUT: Expiration hours
13B		Reserved for future use
• **Extended Section for SPL Version 3.1**		
13C-13F	SPLXTKN	TKN value in return SPL
140-143		Reserved for future use
	SPLEOPST	Possible start of OPTBs
	SPLTLEN	Total length of SPL (X'144')
144	SPLEOPTB	Start of OPTB area
• **VSE/POWER Command Section** The following section is an overlay of the last six sections and defines the command area used when passing a free-format command to VSE/POWER.		
30-B1	SPLCFLD	Command field: max. 130 bytes, terminated by one blank (X'40')
• **User Data in XPCCB Changed by VSE/POWER** The return and feedback codes of bytes 4 and 5 are explained in more detail in Chapter 14, "Return and Feedback Codes and Their Meanings," on page 297.		
00	PXPUSER	DSECT definition
00	PXPBTYP	Buffer type
	PXPBTSPL	X'01' - Spool parameter list
	PXPBTNDB	X'02' - Normal data buffer

Bytes Hex	Field Label	Description/Function
	PXPBTMSG	X'03' - Message buffer
	PXPBTCTL	X'04' - Control record buffer
	PXPBTOPT	X'05' - Buffer with OPTBs
01	PXPACT1	Action type 1
02		Reserved
01-02	PXPLC12	or: Bytes 1 and 2 of data record number returned after restart to active record.
03	PXPINFO	User information byte
	PXPIMSG	X'80' - Message(s) queued
	PXPIORD	X'40' - Order pending
	PXPIPSH	X'20' - VSE/POWER is in shutdown
04	PXPRETCD	Return code
	PXPRCOK	X'00' - No error
	PXPRCOKF	X'04' - Request not handled
	PXPRCERR	X'08' - Request rejected
	PXPRCPVL	X'0C' - Protocol violated or severe error
	PXPRCNOC	X'10' - Connection terminated
05	PXPFBKCD	Feedback code
	PXP00OK	X'00' - No error
	PXP00EOD	X'01' - End of data
	PXP00NJB	X'02' - Job not on job boundary
	PXP00NRS	X'03' - No record spooled
	PXP00RTR	X'04' - Record exceeds maximum specified length
	PXP00ZBF	X'05' - Zero data buffer

Bytes Hex	Field Label	Description/Function
	PXP00CIA	X'06' - Checkpoint identification altered
	PXP00NCM	X'07' - No job completion message retrieval available
	PXP00LCM	X'08' - Only 2 to 5 job completion messages to queue
	PXP00OCM	X'09' - Only 0 to 1 job completion message to queue
	PXP04NOF	X'01' - Job/output not found
	PXP04JOP	X'02' - Job/output protected
	PXP04BSY	X'03' - Job/output marked as active
	PXP04NDS	X'04' - Job/output is not dispatchable
	PXP04IDP	X'05' - Append error, invalid disposition
	PXP04RER	X'06' - Restart error, outside range
	PXP04CER	X'07' - Checkpoint error, outside range
	PXP04SOD	X'08' - Short on spool file space
	PXP04SOA	X'09' - Short on account file space
	PXP04BER	X'0A' - Request not allowed in browse mode
	PXP04DNF	X'0B' - Nothing found while performing a display queue
	PXP04TQN	X'0C' - Temporary queue set not found
	PXP04NMU	X'0D' - No matching user id
	PXP04WDP	X'0E' - RESTART disposition is not D, H, K, L or X
	PXP04JSR	X'0F' - Job suffix number is mandatory
	PXP04NOQ	X'10' - No order/signal queued
	PXP04ONF	X'11' - OPTB(s) not found
	PXP04NJC	X'12' - Job completion message retrieval not available for GCM-OPEN request
	PXP04CKN	X'13' - Extended checkpoint information does not exist

Bytes Hex	Field Label	Description/Function
	PXP04CKE	X'14' - External checkpoint information lost due to I/O error
	PXP04NCK	X'15' - Checkpoint information does not exist
	PXP04NMF	X'16' - No job completion message found (GCM-OPEN)
	PXP04SAC	X'17' - Spool access security violation
	PXP04NAT	X'18' - Queue entry not active or active on other shared system
	PXP04ANS	X'19' - Queue entry active by task not suitable for restart to active record
	PXP04RIS	X'1A' - Restart to active record specified inconsistently together with positioning on line, page, or end of queue entry
	PXP04NRU	X'1B' - Restart to active record request rejected, requestor not in browse mode
	PXP08SPL	X'01' - Invalid SPL
	PXP08REQ	X'02' - Unknown request type
	PXP08SRQ	X'03' - Unknown subrequest type
	PXP08FB2	X'04' - Unknown function byte 2
	PXP08JNM	X'05' - Invalid job name
	PXP08QID	X'06' - Invalid queue identifier
	PXP08CLS	X'07' - Invalid class
	PXP08PWD	X'08' - Invalid password
	PXP08UID	X'09' - Invalid user/remote identifier
	PXP08RFM	X'0A' - Invalid record format
	PXP08DSP	X'0B' - Invalid local or transmission disposition
	PXP08PRY	X'0C' - Invalid priority
	PXP08SID	X'0D' - Invalid system identifier
	PXP08TNN	X'0E' - Invalid destination node name

Bytes Hex	Field Label	Description/Function
	PXP08TUN	X'0F' - Invalid destination user/remote id.
	PXP08FNO	X'10' - Invalid forms identifier
	PXP08FCB	X'11' - Invalid FCB name
	PXP08UCB	X'12' - Invalid UCB name
	PXP08UOP	X'13' - Invalid UCB options
	PXP08FLH	X'14' - Invalid flask identifier
	PXP08CPT	X'15' - Invalid compaction table name
	PXP08CGP	X'16' - Invalid copy groupings
	PXP08CHR	X'17' - Invalid character tables
	PXP08MOD	X'18' - Invalid copy modification tables
	PXP08CCR	X'19' - Invalid characters for copy modification
	PXP08BTS	X'1A' - Buffer too small
	PXP08IAO	X'1B' - Wrong specification of append or restart option
	PXP08IAB	X'1C' - Invalid action request
	PXP08ICR	X'1D' - Invalid control record
	PXP08PRG	X'1E' - Invalid programmer name
	PXP08ROO	X'1F' - Invalid room number
	PXP08DPT	X'20' - Invalid department number
	PXP08BLD	X'21' - Invalid building number
	PXP08CON	X'22' - Conflicting specifications (see also PXPFBKC2)
	PXP08ROL	X'23' - Received record is too large
	PXP08IBT	X'24' - Invalid buffer type
	PXP08ROS	X'25' - Request out of sequence (see also PXPFBKC2)

PWRSPL DSECT

Bytes Hex	Field Label	Description/Function
	PXP08SOS	X'26' - SPL received out of sequence
	PXP08BOS	X'27' - Received buffer is out of sequence
	PXP08RPH	X'28' - Request not allowed
	PXP08ISS	X'29' - Invalid signal specification or signal out of sequence
	PXP08RPW	X'2A' - Record prefix wrong
	PXP08FB1	X'2B' - Unknown function byte 1
	PXP08IML	X'2C' - Invalid record length specified in the SPL
	PXP08IEX	X'2D' - Invalid subsystem name
	PXP08SPA	X'2E' - Complete record is not in the buffer
	PXP08ICC	X'2F' - Invalid carriage control character
	PXP08IOR	X'30' - Invalid order
	PXP08JNO	X'31' - Invalid job number (=0)
	PXP08JSF	X'32' - Invalid job suffix number (>127)
	PXP08IUI	X'33' - Invalid user information
	PXP08IPD	X'34' - GET SPL from RDR queue or PUT SPL not allowed for a DST task
	PXP08UXR	X'35' - Unexpected response received
	PXP08WOS	X'36' - Wait for order out of sequence
	PXP08NSP	X'37' - Invalid separator pages/cards
	PXP08IRR	X'38' - Invalid request for RDR
	PXP08IOP	X'39' - Invalid OPTB specified
	PXP08OLM	X'3A' - OPTB length mismatch
	PXP08DOP	X'3B' - Duplicate OPTBs specified
	PXP08OTL	X'3C' - Specified OPTBs too long

Bytes Hex	Field Label	Description/Function
	PXP08IDH	X'3D' - Invalid DSHR found
	PXP08DIS	X'3E' - Invalid distribution code
	PXP08INK	X'3F' - Invalid keyword OPTB (syntax)
	PXP08NDK	X'40' - Define statement missing for keyword OPTB
	PXP08IDV	X'41' - Invalid value of keyword OPTB
	PXP08CKZ	X'42' - Length of extended checkpoint information is zero
	PXP08CKL	X'43' - Length of extended checkpoint information is too large
	PXP08IQN	X'44' - Queue entry number invalid
	PXP08GJN	X'45' - Generic job name can not be used
	PXP08SEU	X'46' - z/VSE security user id invalid
	PXP08SEP	X'47' - z/VSE security password invalid
	PXP08IPM	X'48' - Incorrect processing mode for PUT-OPEN-OUTPUT
	PXP08IEM	X'49' - PUT SPL with invalid expiration value
	PXP08SDU	X'4A' - GET service: Modify-OPTB rejected for master or duplicate
	PXP08RDU	X'4B' - PUT-OPEN-RESTART rejected for master or duplicate
	PXP08XUA	X'4C' - GCM-XEM service is unavailable or can net be started for application (see also PXPFBKC2)
	PXP0CINS	X'01' - SEND issued, but SENDR required
	PXP0CIXF	X'02' - Used an unsupported XPCC function
	PXP0CBTL	X'03' - Buffer too large
	PXP0CPER	X'04' - Protocol error
	PXP0CPVD	X'05' - Protocol violation by a DDS. Order queued flag not honored
	PXP0CIOE	X'07' - I/O error on either the queue or data file
	PXP0CSNF	X'08' - No VSE/POWER section found in JHR or DSHR

<table>
<tr><th>Bytes Hex</th><th>Field Label</th><th>Description/Function</th></tr>
<tr><td></td><td>PXP0CCOR</td><td>X'09' - Invalid length field in JHR or DSHR</td></tr>
<tr><td></td><td>PXP10CAA</td><td>X'03' - Connection already active</td></tr>
<tr><td></td><td>PXP10PSP</td><td>X'05' - PSTOP issued by operator or due to exit failure</td></tr>
<tr><td></td><td>PXP10SIE</td><td>X'06' - Severe internal error</td></tr>
<tr><td></td><td>PXP10MST</td><td>X'07' - SAS task limit reached</td></tr>
<tr><td>06-07</td><td>PXPROFF</td><td>Offset to invalid record</td></tr>
<tr><td>06-07</td><td>PXPRBLN</td><td>or: Required buffer length</td></tr>
<tr><td>06-07</td><td>PXPLEMC</td><td>or: GCM - Count of lost job event and output generation messages</td></tr>
<tr><td>06-07</td><td>PXPLC34</td><td>or: Bytes 3 and 4 of data record number returned after
restart to active record</td></tr>
<tr><td>06</td><td>PXPFBKC2</td><td>or: Feedback-2 code, valid for RETCD 04, FBKCD 01 and 0B only:</td></tr>
<tr><td></td><td>PXPC2OK</td><td>X'00' - ALL-CMDS, no error</td></tr>
<tr><td></td><td>PXPC2TEM</td><td>X'01' - R|H-CMD no access to DISP=X|A|Y</td></tr>
<tr><td></td><td>PXPC2NOH</td><td>X'02' - H-CMD HOLD only for DISP=D|K</td></tr>
<tr><td></td><td>PXPC2NOR</td><td>X'03' - R-CMD RELEASE only for DISP=H|L</td></tr>
<tr><td></td><td>PXPC2NTA</td><td>X'04' - A-CMD warning nothing to change</td></tr>
<tr><td></td><td>PXPC2CPO</td><td>X'05' - A-CMD COPY change for '*' entry but additional operands given</td></tr>
<tr><td></td><td>PXPC2CDI</td><td>X'06' - A-CMD COPY change for '*' entry but 'PDIR' outbound task found</td></tr>
<tr><td></td><td>PXPC2CNT</td><td>X'07' - A-CMD COPY change for '*' entry, no suitable active task found</td></tr>
<tr><td></td><td>PXPC2BAD</td><td>X'08' - ALL-CMDS|GET, queue record not accessible due to I/O error</td></tr>
<tr><td></td><td>PXPC2FRE</td><td>X'09' - ALL-CMDS|GET, queue rec. empty, already in free Q-record chain</td></tr>
<tr><td></td><td>PXPC2MQU</td><td>X'0A' - ALL-CMDS|GET, mismatch queue</td></tr>
<tr><td></td><td>PXPC2MJM</td><td>X'0B' - ALL-CMDS|GET, mismatch job name</td></tr>
<tr><td></td><td>PXPC2MJB</td><td>X'0C' - ALL-CMDS|GET, mismatch job number</td></tr>
</table>

Bytes Hex	Field Label	Description/Function
	PXPC2IPW	X'0D' - A|H|L|R-CMD, SPL specified user password mismatching Q-rec pwd
	PXPC2BPW	X'0E' - A|H|L|R-CMD, default SPL pwd. No match to Q-record password
	PXPC2JFR	X'0F' - A|H|L|R-CMD job only, FROM-NODE or FROM-USER not matching own
	PXPC2OT1	X'10' - A|H|L|R-CMD,output only,TO-USER not matching to own USER-ID
	PXPC2OT2	X'11' - A|H|L|R-CMD, similar PXPC2OT1
	PXPC2OT3	X'12' - A|H|L|R-CMD, similar PXPC2OT1
	PXPC2OTN	X'13' - A|H|L|R-CMD output only,TO-NODE not matching to own node name
	PXPC2MJS	X'14' - A|H|L|R-CMD|GET, mismatching job(output) suffix
	PXPC2MCL	X'15' - GET-RQ, mismatching job class
	PXPC2MSY	X'16' - GET-RQ, mismatch target sysid
	PXPC2MFU	X'17' - GET-RQ, userid not matching to 'FROM'-userid of job entry
	PXPC2MFT	X'18' - GET-RQ, userid not matching to FROM|TO-userid of output entry
	PXPC2SAC	X'19' - GET/CTL RQ, security logon user ID not equal origin/target user IDVSE/POWER has been started with Spool Access Protection active, the given spool entry does not specify SECAC=NO, and an XPCC program SAS GET/CTL (direct) attempted to access a spool entry. However, either: - the program's security logon user ID (either from the IBM component terminal logon or the partition // ID or * $$ JOB SEC= statement) does not match the spool entry's authorized access user ID(s) (either the spool entry's origin user ID or target user ID), or - the spool entry specifies a target user ID of 'ANY' and the program does not have a security logon user ID The authorized access user ID(s) can be displayed with the PDISPLAY command (displayed as FROM= or TO=).
	PXPC2INC	X'1A' - ALL-CMDS/GET, queue record incomplete - in creation
	PXPC2DEL	X'1B' - GET/CTL-RQ for queue entry in delayed deletion
	PXPC2NVT	X'1C' - GET-RQ, queue entry is either in creation on another system or is being created by a task not eligible to browse an entry in creation. Valid task: execution writer

Bytes Hex	Field Label	Description/Function
	PXPC2EMP	X'1D' - GET-RQ, queue entry is in creation but still empty
	PXPC2QCL	X'1E' - GET-RQ, queue entry is complete in LST queue, no longer in creation
	PXPC2QCP	X'1F' - GET-RQ, queue entry is complete in PUN queue, no longer in creation
	PXPC2QCR	X'20' - GET-RQ, queue entry is complete in RDR queue, no longer in creation
	PXPC2QCX	X'21' - GET-RQ, queue entry is complete in XMT queue, no longer in creation
	PXPC2EXC	X'22' – ALL-CMDS/GET, queue record is not accessible due to "excluded" duplicate
06	PXPFBKC2	or: Feedback-2 code, valid for RETCD 08 and FBKCD 22 only:
	PXPC222A	X'01' - Buffer length is zero, but a buffer type (PXUBTYP) is set.
	PXPC222B	X'02' - Buffer length is zero and no action (PXUACT1) is set and no DST task is running.
	PXPC222C	X'03' - Buffer length is zero and no action (PXUACT1) and no signal (PXUSIGNL) is set and a DST task is running.
	PXPC222D	X'04' - Buffer length is not zero, but no buffer type (PXUBTYP) is set.
	PXPC222E	X'05' - Buffer length is not zero, but a signal (PXUSIGNL) is set and a DST task is running.
	PXPC222F	X'06' - The buffer length is not zero, no (PUT-, GET-, CTL-, GCM-) service is in progress, but a buffer type (PXUBTYP) and an action (PXUACT1) is set.
	PXPC222G	X'07' - The buffer length is not zero, a GET- or a CTL- or a GCM- service is in progress, and a buffer type (PXUBTYP) and an action (PXUACT1) is set.
	PXPC222H	X'08' - A PUT-close request is received and the buffer type is not zero and does not indicate an SPL nor a data buffer.
	PXPC222I	X'09' - A PUT-segment request is received and the buffer type is not zero and does not indicate an SPL nor a data buffer.

Bytes Hex	Field Label	Description/Function
	PXPC222J	X'0A' - A PUT-close-append request is received and the buffer type is not zero and does not indicate an SPL nor a data buffer.
	PXPC222K	X'0B' - A checkpoint request during a PUT service is received and the buffer type is not zero and does not indicate a data buffer.
	PXPC222L	X'0C' - A quit request during a PUT service is received and the buffer type is not zero and does not indicate a data buffer.
06	PXPFBKC2	or: Feedback-2 code, valid for RETCD 08 and FBKCD 25 only:
	PXPC225A	X'01' - The buffer length is zero, no (PUT-, GET-, CTL-, GCM-) service is in progress and none of the following is received: 1. no valid request for message retrieving (PXUACT1 is not PXUATRMR not PXUATABR) 2. no return order request for a DST task 3. no waiting for a order request for a DST task 4. no signal for a DST task
	PXPC225B	X'02' - GET service is in progress and a send data request is received (PXUACT1), but no more data are available.
	PXPC225C	X'03' - Message retrieving is in progress and a return message request is received (PXUACT1), but no more messages are available.
	PXPC225D	X'04' - A GCM service has ended and no new SPL is received (PXUBTYP is not PXUBTSPL).
	PXPC225E	X'05' - A GCM-OPEN-KEEP is being processed and received request (PXUACT1) is not a GCM-MORE nor a GCM-REMOVE request.
	PXPC225F	X'06' - A GCM-OPEN-DELETE is being processed and received request (PXUACT1) is not contain a GCM-MORE request.
	PXPC225G	X'07' - A GCM-OPEN-REMOVE or GCM-OPEN-PURGE is being processed and a request is received (PXUACT1 not 0).
	PXPC225H	X'08' - XEM: GCM-XEM-OPEN or XEM-STOP received prior to GCM-XEM-START
	PXPC225I	X'09' - XEM: GCM-MORE request received but EOD was signaled
6	PXPFBKC2	or: Feedback-2 code, valid for RETCD 08 and FBKCD 4C only:
	PXPC24CA	X'01' - XEM service is unavailable due to insufficient storage for XEM control block
	PXPC24CB	X'02' - Maximal number of running XEM applications is exceeded
	PXPC24CC	X'03' - XEM service already started for Appl-ID
	PXPC24CD	X'04' - No sufficient storage above 16M for application messages queue

PWRSPL DSECT

Bytes Hex	Field Label	Description/Function
	PXPUSLN	Length of control block
• **User Data in XPCCB Changed by User**		
	PXUUSER	DSECT name
00	PXUBTYP	buffer type
	PXUBTSPL	X'01' - Spool parameter list
	PXUBTNDB	X'02' - Normal data buffer
	PXUBTCTL	X'04' - Control record buffer
01	PXUACT1	Action type 1
	PXUATEOD	X'01' - End of data (PUT function)
	PXUATRQS	X'02' - Close queue entry (GET function)
	PXUATABR	X'03' - Quit request
	PXUATSGM	X'04' - Segmentation request
	PXUATROE	X'05' - End of data for appendable output
	PXUATPRG	X'06' - Purge queue entry request
	PXUATCHK	X'07' - Checkpoint request
	PXUATRMR	X'08' - Return message request
	PXUATSDR	X'09' - Send data request
	PXUATFLH	X'0A' - Flush hold request
	PXUATROR	X'0B' - Return order/signal immediately
	PXUATWFR	X'0C' - Wait till order/signal to return
	PXUAT1PF	X'0D' - Printing/punching failed
	PXUATCKR	X'0E' - Retrieve external checkpoint information
	PXUATDEL	X'10' - Delete retrieved messages (GCM-REMOVE)

Bytes Hex	Field Label	Description/Function
	PXUATGCM	X'11' - Retrieve more messages (GCM-MORE)
02		Reserved
03	PXUINFO	User information byte
04	PXURETCD	Return code
05	PXUFBKCD	Feedback code
06	PXUSIGNL	Signal byte
	PXUSDSTP	X'01' - Device stopped
	PXUSSET	X'02' - Setup processed
07		Reserved
	PXUUSLN	Length of control block
• **VSE/POWER Record Prefix Layout**		
	RECPRFIX	DSECT name
00	RECCCODE	Command code
01	RECTYPE	Record type
	RECTNORM	X'00' - Normal data record Used for all records of RDR queue or normal (non-CPDS) data records of LST/PUN queue. Normal when data record prefix in DBLK reflects 'line print or card move data'.
	RECTSPL	X'01' - Spool parameter list
	RECTFIXM	X'02' - Fixed format message
	RECTSEPR	X'03' - Start separator page/card record
	RECT3540	X'04' - 3540 data record
	RECTCCR	X'05' - Control command record Control when data record prefix in DBLK does not reflect 'line print or card move data'.
	RECTCPDS	X'06' - CPDS data record

PWRSPL DSECT

Bytes Hex	Field Label	Description/Function
	RECTESEP	X'07' - End separator page/card record
	RECTEOC	X'08' - End of copy record
	RECTFJCM	X'09' - GCM: Fixed format job completion message
	RECTFJGM	X'0A' - GCM: Fixed format job generation message
	RECTFOGM	X'0B' - GCM: Fixed format output generation message
	RECTFXEM	X'0C' - GCM: Fixed format extended event message
02-03	RECLNGTH	Logical record length (excluding 8-byte prefix)
04-07	RECLOGNO	Logical record number
	RECPRFXL	Record prefix length

Bytes Hex	Field Label	Description/Function
• **VSE/POWER Fixed Format Queue Display Record**		
	PXFMDSCT	DSECT name
00-01	PXFMRLEN	Record length
02	PXFMTYPE	Record type
	PXFMTQDI	X'01' - Fixed format queue display
03	PXFMVOL	BAM tape volume number
	PXFMVOLA	X'80' - Last-volume flag X'01'-X'7E' - Volumes 1-126 X'7F' - Volume 127 or higher
04-0B	PXFMDATE	Creation date of queue entry (mm/dd/yy or dd/mm/yy) For creation date century, see PXFMDATC
0C-0F	PXFMSTRT	Start time (0HHMMSSF), packed format
10-13	PXFMSTOP	Stop time (0HHMMSSF), packed format
14-23	PXFMUSER	User information
24-2B	PXFMNAME	Job name

Bytes Hex	Field Label	Description/Function
2C-2D	PXFMJNUM	Job number
2E	PXFMJSUF	Job suffix - X'00' to X'7F' (0 to 127)
	PXFMJSLA	X'80' - Last segment indication in bit 0 Actual segment number 1 - 127 in bit 1 - 7
2F	PXFMQUID	Queue identifier
30	PXFMCLSS	Class
31	PXFMPRIO	Priority
32	PXFMDISP	Disposition ('*' = in execution by task with update authority)
33	PXFMCOPY	Number of copies
34	PXFMFLG1	Control flag 1
	PXFMF1XQ	X'80' - Queue set resides in the XMT queue
	PXFMF1AB	X'40' - Abended queue set, DISP=X
	PXFMF1AP	X'20' - Appendable queue set, DISP=A
	PXFMF1CP	X'10' - Checkpointed queue set in-creation queue
	PXFMF1PF	X'08' - Printing/punching failed, DISP=Y
	PXFMF1EX	X'04' - Due date expired
	PXFMF1SE	X'02' - Job is authenticated
	PXFMF1SA	X'01' - Not spool access protected
35	PXFMRCFM	Record format
36	PXFMSTAT	Paper status byte
		C'B' - burst requested
37	PXFMSYID	Target or processing system identifier, or 'M', if parallel browsing

Bytes Hex	Field Label	Description/Function
38-3B	PXFMREC#	Number of records spooled. If RDR-type (I=R) entry in the XMT queue with DISP=*, then "remaining" number of records to be transmitted.
3C-3F	PXFMPGE#	Number of pages spooled. IF LST queue entry with DISP=*, then "remaining" number of pages to be printed. Note - no remaining pages shown if in the XMT queue.
40-43	PXFMLNE#	Number of lines/cards spooled. If LST/PUN queue entry or LST/PUN-type (I=L or I=P) in the XMT queue, then "remaining" number of lines/cards to be printed/punched or to be tramsmitted.
44-47	PXFMFLSH	Flash identifier
48-4F	PXFMFORM	Forms identifier (FNO)
50-57	PXFMCPYG	Copy groupings
58	PXFMFLG2	Control flag 2
	PXFM2SDF	X'80' - Class defined as static
	PXFM2SRN	X'40' - Static class running
	PXFM2SWW	X'20' - Static class waiting for work
	PXFM2DDF	X'10' - Class defined as dynamic
	PXFM2DSP	X'08' - Dynamic class suspended
	PXFM2DEN	X'04' - Dynamic class enabled
	PXFM2PRP	X'02' - In execution preparation phase
	PXFM2RUN	X'01' – NORUN=IGN specified in the * $$ JOB statement
59	PXFMNSEP	Number of separator cards or pages
5A-5B	PXFMJBO#	Original job number
5C-5F	PXFMCMPT	Compaction table name
60-67	PXFMNODE	Target destination node name
68-6F	PXFMUSID	Target destination user/remote id

Bytes Hex	Field Label	Description/Function
70-77	PXFMORGN	Originating node name
78-7F	PXFMORGU	Originating user/remote id
80-87	PXFMSUBS	Subsystem name (external writer id)
88-8C	PXFMDDND	Next due date
88-89	PXFMDDN1	Day or month
8A	PXFMDDS1	Separator '/'
8B-8C	PXFMDDN2	Day or month
8D-91	PXFMDDNT	Next due time
8D-8E	PXFMDDNH	Hours
8F	PXFMDDS2	Separator
90-91	PXFMDDNM	Minutes
92-93	PXFMDATC	Century (cc) of creation date at PXFMDATE
94-97	PXFMQNUM	Queue entry number, binary
98-9F	PXFMSECN	Job security zone if job is authenticated
A0-A7	PXFMDIST	Output distribution code
A8-B1	PXFMMACC	Multiple browse access counts:
A8	PXFMMACN	Non-shared access count
A9	PXFMMAC1	Shared SYSID 1 access count
AA	PXFMMAC2	Shared SYSID 2 access count
AB	PXFMMAC3	Shared SYSID 3 access count
AC	PXFMMAC4	Shared SYSID 4 access count
AD	PXFMMAC5	Shared SYSID 5 access count
AE	PXFMMAC6	Shared SYSID 6 access count

Bytes Hex	Field Label	Description/Function
AF	PXFMMAC7	Shared SYSID 7 access count
B0	PXFMMAC8	Shared SYSID 8 access count
B1	PXFMMAC9	Shared SYSID 9 access count
B2	PXFMFLG3	Control flag 3
	PXFM3CRE	X'80' - Queue entry is in creation
	PXFM3DEL	X'40' - Queue entry is in deletion
	PXFM3PLI	X'20' – Queue entry is in creation, PUN into AF library
B3	PXFMMDUP	Number of duplicates (entry is a master queue entry)
B4-BA	PXFMTASK	Task descriptor of task creating this queue entry
BB	PXFMOWNT	Task type of task creating this queue entry
	PXFMOWNJ	'J' - queue entry being created by JOB
	PXFMOWNN	'N' - queue entry being created by Networking
	PXFMOWNR	'R' - queue entry being created by Remote Station
	PXFMOWNS	'S' - queue entry being created by SAS Application
	PXFMOWNB	' ' - queue entry being created by other task
BC-C3	PXFMOWND	Task type of task creating this queue entry
C4-CF	PXFMEDY	Expiration moment:
C4-C5	PXFMEDYA	Month or day
C6-C7	PXFMEDYB	Day of month
C8-CB	PXFMEDYY	Year
CC-CD	PXFMEDYH	Hour
CE-CF	PXFMEDYM	Minute
D0-D3	PXFMMNUM	Queue entry number of master, entry is duplicate

Bytes Hex	Field Label	Description/Function
D4-D7	PXFMTKN	TKN value in HEX
D8-EF		Reserved for future use
	PXFMLENG	Length of DSECT
• **VSE/POWER Restart Control Record (Layout)**		
	PXRSDSCT	DSECT name
00-01	PXRSRLEN	Length of the restart record
02	PXRSTYPE	Record type
	PXRSTRST	X'02' - Restart control record
03		Reserved
04-07	PXRSRECN	Logical record number specifying where to begin the restart
08	PXRSRCPY	Associated restart copy number
09	PXRSOPT	Option byte
	PXRSOPOL	X'80' - Positioning on line requested (ignored for RDR/PUN type queue entries)
	PXRSOPAE	X'40' - Positioning at end of queue entry requested if restart number is too high
	PXRSOPOP	X'20' - Positioning on page requested (ignored for RDR type queue entries)
	PXRSOPAR	X'10' - Positioning on active record requested
0A-0B		Reserved
	PXRSLENG	Length of DSECT
• **VSE/POWER Checkpoint Control Record (Layout)**		
	PXCPDSCT	DSECT name
00-01	PXCPRLEN	Length of checkpoint record
02	PXCPTYPE	Record type

Bytes Hex	Field Label	Description/Function
	PXCPTCHK	X'03' - Checkpoint control record
03	PXCPFLAG	Flag byte
	PXCPFXIE	X'80' - Checkpoint control record with extended information
04-07	PXCPRECN	Logical record number of the checkpoint record
08	PXCPRCPY	Associated copy number
09-0B		Reserved
	PXCPLENG	Length of DSECT
0C	PXCPSTXI	Start of checkpoint with extended information (length from 1 to DBLK size minus 288)
• **VSE/POWER Checkpoint Response Control Record (Layout)**		
	PXCRDSCT	DSECT name
00-01	PXCRRLEN	Length of checkpoint response record
02	PXCRTYPE	Record type
	PXCRTCRS	X'04' - Checkpoint response control record
03	PXCRFLAG	Flag byte
	PXCRFXIE	X'80' - Checkpoint with extended information exists
	PXCRFXIS	X'40' - Checkpoint with extended information saved
04-0B	PXCRJNAM	Job name
0C-0D	PXCRJNUM	Job number
0E	PXCRJSUF	Job suffix - X'00' to X'7F' (0 to 127)
	PXCRJSLA	X'80' - Last segment indication in bit 0 Actual segment number 1 - 127 in bit 1 - 7
0F	PXCRRCPY	Associated copy number
10-13	PXCRRECN	Logical record number associated with the checkpoint

Bytes Hex	Field Label	Description/Function
14-17	PXCRQNUM	Associated queue entry number
18-1B		Reserved for future use
	PXCRLENG	Length of DSECT
1C	PXCRSTXI	Start of checkpoint with extended information
• **VSE/POWER GET-OPTB Control Record**		
	PXGODSCT	DSECT name
00-01	PXGORLEN	Length of control record
02	PXGOTYPE	Record type
	PXGOTGOP	X'08' - GET-OPTB control record
03		Reserved for future use
04-05	PXGOID	OPTB identifier (0 for all)
• **VSE/POWER MODIFY-OPTB Control Record**		
	PXMODSCT	DSECT name
00-01	PXMORLEN	length of control record
02	PXMOTYPE	record type
	PXMOTMOP	X'08' - MODIFY-OPTB control record
03		Reserved for future use
04-07		Reserved for future use
08	PXMOOPTB	Output parameter text block
• **DSECT for SPL version 1 and 2**		
	XTSOVSPL	DSECT name
00-2F	XTSGSECT	General section part 1
30-A5	XTSDSECT	General section part 2

Bytes Hex	Field Label	Description/Function
A6-CF	XTSOSECT	Output section
D0-F7	XTS3SECT	3800 section
F8-FB	XTSESECT	OPTB section
F8-F9	XTSOPOF	Offset to OPTBs
FA-FB	XTSOPLN	Length of OPTBs
FC	XTSEOPTB	Possible start of OPTBs
• **VSE/POWER Order Control Record** The order control record consists of two sections: - the header section - the variable order section		
	PORDER	DSECT name
00-01	PORDRLEN	Length of order
02	PORDTYPE	Record type
	PORDREC	X'05' - Order control record
03	PORDMOD	Order request type
	PORDMSTR	X'01' - Start device order
	PORDMSTP	X'02' - Stop device order
	PORDMRST	X'03' - Restart device order
	PORDMPGO	X'04' - Re-activate device order
	PORDMSET	X'05' - Setup device order
	PORDMFLH	X'06' - Flush device order
	PORDMXMT	X'07' - User defined order
	PORDMSND	X'10' - Send message order
	PORDMSLD	X'11' - Set logical destination order

Bytes Hex	Field Label	Description/Function
	PORDMPAO	X'12' - Account record order
04	PORDFLAG	Flag byte
	PORDFQFD	X'80' - Queue for display
05	PORDMSGL	Length of message
06-07	PORDAFPL	Length of Advanced Function Printing account record
08-1F	PORDDEST	Destination for order
08-0F	PORDSUBS	Requesting subsystem identifier
10-17	PORDNODE	Requesting node name
18-1F	PORDUSER	Requesting user identifier
	PORDHLEN	Length of header section
• **VSE/POWER Start Device Order Section**		
20-27	PORDSDEV	Device name
28-2B	PORDSCLS	Class(es)
2C-2D		Reserved
2E	PORDSFLG	Flag byte
	PORDSSKP	X'80' - PSTART with SKIP=YES
2F	PORDSPSL	Length of parameter string
30-6B	PORDSPRM	Parameter string
	PORDSLEN	Length of start device order
• **VSE/POWER Stop Device Order Section**		
20	PORDPTRB	Termination request byte
	PORDPEOJ	X'80' - Terminate at end of job
	PORDPIMM	X'40' - Terminate immediately

PWRSPL DSECT

Bytes Hex	Field Label	Description/Function
	PORDPRST	X'20' - Terminate with restart
21-22		Reserved
23	PORDPPSL	Length of parameter string
24-5F	PORDPPRM	Parameter string
	PORDPLEN	Length of stop device order
• **VSE/POWER Setup Device Order Section**		
20-23	PORDUPGE	Number of pages to setup
24-2E		Reserved
2F	PORDUPSL	Length of parameter string
30-6B	PORDUPRM	Parameter string
	PORDGLEN	Length of setup device order
• **VSE/POWER Reactivate Device Order Section**		
20-22		Reserved
23	PORDGPSL	Length of parameter string
24-5F	PORDGPRM	Parameter string
	PORDULEN	Length of reactivate device order
• **VSE/POWER Restart Device Order Section**		
20	PORDTFLG	Flag byte
	PORDTPOS	X'80' - Positive displacement
	PORDTMIN	X'40' - Negative displacement
	PORDTABS	X'20' - Absolute displacement from begin of file
21-23		Reserved
24-27	PORDTPGE	Number of pages/lines for restart

Bytes Hex	Field Label	Description/Function
28-2E		Reserved
2F	PORDTPSL	Length of parameter string
30-6B	PORDTPRM	Parameter string
	PORDTLEN	Length of restart device order
• **VSE/POWER Flush Device Order Section**		
20	PORDFFLG	Flag byte
	PORDFHLD	X'80' - Flush hold requested
21-22		Reserved
23	PORDFPSL	Length of parameter string
24-5F	PORDFPRM	Parameter string
	PORDFLEN	Length of flush device order
• **VSE/POWER Xmit Device Order Section**		
20	PORDXPSL	Length of command string
21-A4	PORDXPRM	Parameter string
	PORDXLEN	Length of xmit device order
• **VSE/POWER Send Message Order Section (inbound)**		
20-97	PORDMMSG	Message text
	PORDMLEN	Length of send message order section
• **VSE/POWER Set Logical Destination Order Section (inbound)**		
20-5F	PORDLOG8	Eight logical destination names
20-27	PORDDLOG	Logical destination name
	PORDDLEN	Length of set logical destination order section
• **VSE/POWER Put Account Record Order (inbound)**		

PWRSPL DSECT

Bytes Hex	Field Label	Description/Function
20-..	PORDAFPA	Start of Advanced Function Printing account record
• **VSE/POWER Order Response Control Record**		
	PORDRESP	DSECT name
00-01	PORSRLEN	Record length
02	PORSTYPE	Record type
	PORSREC	X'06' - Order response control record
03	PORSMOD	Order request type. See order control record for definitions of order types.
04	PORSFLAG	Flag byte
	PORSFMID	X'80' - PORSMID contains 4-byte message-id
05	PORSMSGL	Length of message
06-07	PORSRCFC	Return code and feedback code
06	PORSRETC	Return code
	PORSROK	X'00' - Order accepted
	PORSROKF	X'04' - Order accepted, but request can not be handled
	PORSRINV	X'08' - Order invalid or not accepted
07	PORSFDBK	Feedback code
	PORSFOK	X'00' - Order accepted and valid.
Feedback Code from the User to VSE/POWER		
	PORSFPAR	X'01' - Parm string missing or invalid
	PORSFONA	X'02' - Order not accepted
	PORSFDUN	X'03' - PSTART - device unknown
	PORSFDBS	X'04' - PSTART - device busy
	PORSFDOS	X'05' - PSTART - device out of service

Bytes Hex	Field Label	Description/Function
	PORSFDRJ	X'06' - PSTART - rejected
Feedback Code from VSE/POWER to the User		
	PORSFNAC	X'01' - Accounting not initialized
	PORSFINV	X'01' - Invalid or unknown order
	PORSFOTS	X'02' - Order buffer too small for the passed order control record
	PORSFMSG	X'03' - Message length too large
	PORSFSLD	X'04' - Set logical destination with an invalid destination
	PORSFPAC	X'05' - Mismatch of length fields within order
	PORSFRTL	X'06' - Account record too small (< 55 bytes) or too large (> 1000 bytes)
08-1F	PORSDEST	Destination
08-0F	PORSSUBS	Destination subsystem identification
10-17	PORSNODE	Destination node name
18-1F	PORSUSER	Destination user identification
	PORSHLEN	Length of header section
20-97	PORSMSG	Message text, if order response sent to VSE/POWER
20-23	PORSMID	Message-id, if order response sent to user, and PORSFMID is set
	PORSTLEN	Length of total record
• **VSE/POWER Signal Control Record**		
	PSIGNAL	DSECT name
00-01	PSGNRLEN	Record length
02	PSGNLTYP	Record type
	PSGNLREC	X'07' - Signal control record
03	PSGNLMOD	Signal type

Bytes Hex	Field Label	Description/Function
	PSGNLTOA	X'01' - Output arrived signal
04-07		Reserved
	PSGNLLEN	Length of record
• **VSE/POWER Message Control Record**		
	PMSGREC	DSECT name
00-01	PMSGRLEN	Record length
02	PMSGTYPE	Record type
	PMSGTREC	X'80' - Message control record
03		Reserved
04	PMSGFLAG	Flag byte
05	PMSGTXTL	Message text length
06-07		Reserved
08-0F	PMSGSUBS	Destination subsystem identifier
10-17	PMSGNODE	Destination node name
18-1F	PMSGUSER	Destination user identifier
	PMSGHLEN	Length of header section
20-97	PMSGTEXT	Message text
	PMSGTLEN	Length of total record
• **VSE/POWER Notify Control Record**		
	PNTYREC	DSECT name
00-01	PNTYRLEN	Record length
02	PNTYTYPE	Record type
	PNTYTREC	X'81' - Notify control record

Bytes Hex	Field Label	Description/Function
03		Reserved
04	PNTYFLAG	Flag byte
05-07		Reserved
08-0F	PNTYJNAM	Job name
10-11	PNTYJNUM	Job number
12	PNTYJSUF	Job suffix - X'00' to X'7F' (0 to 127)
	PNTYJSLA	X'80' - Last segment indication in bit 0 Actual segment number 1 - 127 in bit 1 - 7
13	PNTYJCLA	Job class
14-1B	PNTYDEST	Destination user identification
	PNTYLEN	Length of record
• **VSE/POWER Fixed Format Job Completion Message Record**		
	JCMDS	Fixed format JCM record
00-04	JCMID	Message number (1Q5DI)
05-06		reserved
07	JCMFLT	System configuration info
	JCMFDD	X'80' - Date format is ddmmyy
08-0F	JCMFNAM	Job name
10-13	JCMFNUM	Job number
14-17	JCMFONUM	Job number of originating job or hex zero
18-1F	JCMFNOD	Node-id of execution node
20-27	JCMFECT	Execution completion time
28-2B	JCMFLRC	Last return code
2C-2F	JCMFMRC	Maximum return code

Bytes Hex	Field Label	Description/Function
30-37	JCMFECD	Execution completion date (see also JCMFECDC)
38	JCMFJC7	Job Control Switch 7 (JCSW7)
	JCMFJ7CA	X'80' - Operator CANCEL pending
	JCMFJ7JC	X'02' - Job Control cancelation
39	JCMFJC8	Job Control Switch 8 (JCSW8)
	JCMFJ8AB	X'08' - Abnormal termination
3A-43	JCMFDUR	Job duration information
44-45	JCMFECDC	Century of processing completion date JCMFECD
46-4F		Reserved
50	JCMFPRIV	Data from SPLXPRIV
58	JCMFUSID	User-id from SPLGUS valid at job submission time
	JCMFLEN	Length of JCM record
• **VSE/POWER Fixed Format Job Generation Message Record**		
	JGMDS	Fixed format JGM record
00-04	JGMID	Message number (1Q5HI)
05-07		Reserved
08-0F	JGMFNAM	Generating job name
10-13	JGMFNUM	Generating job number
14-1B	JGMFNNAM	Generated job name
1C-1F	JGMFNNUM	Generated job number
20-4B		Reserved
4C-4F	JGMF1NUM	Original job number of generating job when it entered the system for the first time
50-57	JGMFPRIV	Data from SPLXPRIV

Bytes Hex	Field Label	Description/Function
58-5F	JGMFUSID	User-id from SPLGUS
	JGMFLEN	Length of JGM record
• **VSE/POWER Fixed Format Output Generation Message Record**		
	OGMDS	
00-04	OGMID	Message number (1Q5RI)
05-06		Reserved
07	OGMFLT	System configuration information (COMREG)
	OGMFDD	X'80' - date format is DD/MM/YY
08-0F	OGMFNAM	Generating job name
10-13	OGMFNUM	Generating job number
14-1B	OGMFNNAM	Generated output name
1C-1F	OGMFNNUM	Generated output number
20-27	OGMFTIME	Output created time in the packed format HH:MM:SS
28-2F	OGMFNDID	Originator node ID
30-37	OGMFDATE	Output created date in the packed format DD/MM/YY or MM/DD/YY (see OGMFLT)
38	OGMFNSFX	Generated output suffix number (for the segmented output)
39	OGMFQU	Output identifier ('L' for LST and 'P' for PUN)
3A-41	OGMFDNID	Destination node ID for the generated output
42-43		Reserved
44-45	OGMFDATC	Output created date century in the packet format CC
46-4B		Reserved
4C-4F	OGMFONUM	Original generating job number
50-57	OGMFPRIV	Data from SPLXPRIV

Bytes Hex	Field Label	Description/Function
58-5F	OGMFUSID	User-ID from SPLGUS
	OGMFLEN	Length of OGM record
• **VSE/POWER Fixed Format eXtended Event Message Record**		
	XEMDS	
00-04	XEMID	Message Identifier (1Q5XI)
05	XEMEID	Event Identifier
	XEMECR	C'C' – Creation
	XEMEAC	C'A' – Alteration by a command
	XEMEAP	C'P' – Alteration by processing
	XEMEDL	C'D' – Deletion
06	XEMFLG	Flag byte
	XEMFDD	X'80' – Event's date has DD/MM/YY format (from System Configuration Information - COMREG)
	XEMERT	X'40' – Execution Reader Task ID ("E xx" within XEMTSK field)
	XEMFXQ	X'20' – Queue entry in XMT queue
	XEMFCN	X'10' – Queue entry altered/deleted due to PFLUSH | PCANCEL command
	XEMFCI	X'08' – Command processor ('O CP') invoked internally
	XEMFCSC	X'04' – Segment requested by PSEGMENT | PALTER SEGMENT command
		X'02' – Reserved for future use
		X'01' – Reserved for future use
07		Reserved for future use
08-0B	XEMTSK	Task ID
0C-0F	XEMCUU	CUU (Physical device ID)
10-17	XEMDATE	Event date (DD/MM/YY or MM/DD/YY according with the XEMFLG flag)
18-1F	XEMTIME	Event time in the format HH:MM:SS
20-21	XEMDATC	Event date century (CC)
22	XEMQI	Q-record ID (QRQI)
23		Reserved for future use
24-2B	XEMNAM	Queue entry name
2C-2D	XEMNUM	Queue entry number (hexadecimal)
2E	XEMSFX	Queue entry suffix (hexadecimal)
2F	XEMDISP	Queue entry disposition
30	XEMCLS	Queue entry class
31	XEMPRI	Queue entry priority
32-33		Reserved for future use
34-37	XEMQNUM	Queue entry QNUM (hexadecimal)
38-3B	XEMTKN	Queue entry TKN value
3C-4B	XEMUINF	User information (UINF)
4C	XEMC1	Queue entry change bit map 1

Bytes Hex	Field Label	Description/Function
	XEMC1CLS	X'80' – Change class: CLASS
	XEMC1CMP	X'40' – Change performing of data compaction for SNA terminal: CMPACT
	XEMC1CPY	X'20' – Change number of output copies: COPY
	XEMC1DSP	X'10' – Change disposition: DISP
	XEMC1DIS	X'08' – Change output distribution code: DIST
	XEMC1DUE	X'04' – Nullify Due Date: DUETIME=NULL
	XEMC1EMO	X'02' – Change expiration moment: EXPDAYS or EXPHRS or NULL
	XEMC1FCB	X'01' – Change name of FCB image phase: FCB
4D	XEMC2	Queue entry change bit map 2
	XEMC2FNO	X'80' – Change form-number specification: FNO
	XEMC2NOD	X'40' – Change final destination: NODE
	XEMC2PRI	X'20' – Change priority: PRI
	XEMC2REM	X'10' – Change Remote-ID: REMOTE
	XEMC2SID	X'08' – Change system ID: SYSID
	XEMC2INF	X'04' – Change User information: UINF
	XEMC2USR	X'02' – Change User-ID: USER
		X'01' – Reserved for future use
4E	XEMC3	Queue entry change bit map 3
	XEMC3FWR	X'80' – Re-queue from Wait For Run sub-queue (time event)
	XEMC3XTR	X'40' – Transition between local queue (RDR | LST | PUN) and XMT queue (PALTER NODE command)
		X'20' – Reserved for future use
		X'10' – Reserved for future use
		X'08' – Reserved for future use
		X'04' – Reserved for future use
		X'02' – Reserved for future use
		X'01' – Reserved for future use
4F		Reserved for future use
50-57	XEMFUSER	'From' User-ID
58-5F	XEMFNODE	'From' Node-ID
60-67	XEMTUSER	'To' User-ID
68-6F	XEMTNODE	'To' Node-ID
70-77	XEMQDATE	Queue entry creation date
78-7F	XEMQTIME	Queue entry creation time
80-81	XEMQDATC	Queue entry creation century
82-87		Reserved for future use
88-8F	XEMSUSER	SAS User task: User-ID (not used by other tasks)
90-97	XEMSAPPL	SAS User task: Application-ID (not used by other tasks)
98-9F	XEMWGNAM	Execution writer: generating job name (not used by other tasks)
A0-A1	XEMWGNUM	Execution writer: generating job number (not used by other tasks)

PWRSPL DSECT

Bytes Hex	Field Label	Description/Function
A2-F7		Reserved for future use

Chapter 13. Spool-Access Support Programming Example

The sample routine shown here is delivered to you as an A-book in PRD1.MACLIB under the name of PWRSASEX.A . Here follow:

1. The set of statements that causes the source code of the spool-access support example to be assembled, linked, and cataloged.
2. An inline macro definition.
3. The source code, which is provided under "Programming Example Source Code" on page 272 primarily for study and reference purposes.
4. The set of statements that request execution of the sample routine phase 'PWRSASEX'.
5. The console printlog of PWRSASEX execution.

Control Statements for Assembly and Catalog

```
* $$ JOB JNM=PWRSACAT,DISP=D,CLASS=A
// JOB PWRSACAT
// OPTION CATAL
// LIBDEF *,SEARCH=PRD1.MACLIB
// LIBDEF *,CATALOG=...
*
* PROVIDE ... CATALOG LIB.SUBLIB FOR PWRSASEX
*
// EXEC ASSEMBLY,SIZE=100K
         COPY PWRSASEX
         END
/*
// EXEC LNKEDT
/&
* $$ EOJ
```

Use the printout of job PWRSACAT, namely the Assembler listing of program PWRSASEX, for a detailed study of generated macro code and DSECT addresssing.

Inline Macro Definition

This macro definition, which precedes the source code, provides for a display of messages on the system console. Only the beginning and end of the instructions of this definition are shown here.

```
         TITLE PWRSASEX  -  SAS EXAMPLE PROGRAM
         MACRO
&LABEL   DPLAY &LINE,&LENGTH,&ID=1
         GBLB  &FDSP(15)
         LCLA  &LINLEN,&LENLEN;
         LCLC  &LENREG,&LINREG,&DISP;
         LCLB  &LENSW,&LINSW,&TXT,&DEF;
         AIF   (T'&ID EQ 'N' AND &ID LE 15).L001
         MNOTE 8,'ID NOT NUMERIC OR GREATER THAN 15'
         MEXIT
.L001    AIF   (T'&LINE NE 'O').L002
          ... ... ...
          ... ... ...
          ... ... ...
.L018    ANOP
         L     0,=A(&LINE)
.L019    ANOP
         STCM  0,7,DSCCW&ID+1;
```

```
         L     1,=A(DSCCB&ID)
         EXCP  (1)
         WAIT  (1)
         MEND
```

Programming Example Source Code

```
         PUNCH ' PHASE PWRSASEX,*'                                      00124000
******************************************************************** 00125000
**                                                                   ** 00126000
**                      P W R S A S E X                              ** 00127000
**         VSE/POWER SPOOL ACCESS SUPPORT:  EXAMPLE PROGRAM          ** 00129000
**                                                                   ** 00130000
******************************************************************** 00131000
*                                                                     * 00132000
*   THIS PROGRAM - NAMED PWRSASEX - ACTS AS A SPOOL-ACCESS-SERVICE    * 00133000
*   USER THAT INTERACTS WITH VSE/POWER USING THE SPOOL-ACCESS SUPPORT * 00134990
*                                                                     * 00136000
*   PWRSASEX CAN RUN IN ANY PARTITION, UNDER OR OUTSIDE THE CONTROL   * 00137000
*   OF VSE/POWER.  FOR SUCCESSFUL COMPLETION, HOWEVER, AN ADDITIONAL  * 00138000
*   PARTITION UNDER CONTROL OF VSE/POWER AND WITH EXECUTION ...       * 00139290
*                         C L A S S  =  4                             * 00139580
*   MUST BE WAITING FOR WORK.                                         * 00140000
*   N O T E : THE MANUAL SUGGESTS TO SUBMIT JOB 'PWRSARUN' TO THE     * 00140100
*             EXECUTION CLASS=A. PWRSARUN GIVES CONTROL TO 'PWRSASEX' * 00140200
*             THAT SUBMITS JOB 'EXAMPLE' FOR CLASS=4 AND THAT SURVEYS * 00140300
*             THE EXECUTION OF JOB EXAMPLE. YOU MAY CHANGE THE EXE-   * 00140400
*             CUTION CLASS OF JOB EXAMPLE BY ALTERING THE * $$ JOB    * 00140500
*             STATEMENT AT CODE LABEL 'JECL1'. JOB PWRSARUN AND JOB   * 00140600
*             EXAMPLE MAY EVEN HAVE THE SAME DYNAMIC CLASS, PROVIDED  * 00140700
*             THIS CLASS ALLOWS AT LEAST TWO PARTITIONS TO BE ACTIVE. * 00140800
*                                                                     * 00141000
*   THE PROGRAM'S OPERATIONAL STEPS ARE:                              * 00142000
*                                                                     * 00143000
*   1.  IDENTIFY ITSELF TO THE SYSTEM'S XPCC SUPPORT WITH THE USER    * 00144000
*       IDENTIFICATION 'PWRSASEX'.                                    * 00145490
*                                                                     * 00146000
*   2.  TRY TO ESTABLISH A COMMUNICATION PATH TO VSE/POWER -- TERMIN- * 00147000
*       ATE IF THIS PATH CANNOT BE ESTABLISHED WITHIN TWO MINUTES     * 00148000
*                                                                     * 00149000
*   3.  USE THE PUT SERVICE TO SUBMIT THE JOB 'EXAMPLE' TO THE        * 00150000
*       VSE/POWER RDR QUEUE FOR EXECUTION IN CLASS=4.                 * 00151490
*                                                                     * 00152000
*   4.  USE THE CTL SERVICE TO SUBMIT A PDISPLAY COMMAND, IN          * 00153590
*       ORDER TO LOCATE THE OUTPUT OF JOB 'EXAMPLE' IN THE            * 00154180
*       VSE/POWER LST QUEUE, AND SHOW THE QUEUE-DISPLAY MESSAGE ON    * 00155000
*       THE CONSOLE. IF THE OUTPUT IS NOT YET AVAILABLE, THE PROGRAM  * 00156590
*       RE-ISSUES THE PDISPLAY COMMAND EVERY 10TH OF A SECOND FOR TWO * 00157180
*       MINUTES.  IF THEN THE OUTPUT IS STILL NOT AVAILABLE, PWRSASEX * 00158000
*       TERMINATES.                                                   * 00159000
*       N O T E :  THE CTL SERVICE IS PRESENTED IN TWO FLAVOURS, YOU  * 00159300
*                  MAY SELECT FLAVOUR TWO AT CODE LABEL 'CTL1'.       * 00159600
*                                                                     * 00160000
*   5.  RETRIEVE THE LST QUEUE ENTRY 'EXAMPLE' USING THE GET SERVICE. * 00161000
*       N O T E :  THE GET SERVICE IS PRESENTED IN TWO FLAVOURS ACC.  * 00161300
*                  TO THE PRE-SELECTION AT CODE LABEL 'CTL1'.         * 00161600
*                                                                     * 00162000
*       THE PROGRAM CAUSES THE COMPLETE ENTRY TO BE DISPLAYED ON THE  * 00163000
*       CONSOLE. PWRSASEX ISSUES A GET-RESTART REQUEST THAT POSITIONS * 00164490
*       THE RETRIEVAL POINTER IN THE MIDDLE OF THE QUEUE ENTRY AND    * 00165000
*       REDISPLAYS THE SECOND HALF.                                   * 00166000
*                                                                     * 00167000
*       PWRSASEX ENDS GET PROCESSING BY ISSUING A QUIT REQUEST.       * 00168000
*                                                                     * 00169000
*   6.  SUBMIT THE DATA CARDS OF JOB 'EXAMPLE' TO THE VSE/POWER       * 00173000
*       LST-QUEUE AS ENTRY 'EXAMPSEG' AND ISSUE PUT-SEGMENT REQUESTS  * 00174000
```

```
*       TO GET THREE (RBS-LIKE) SEGMENTS OF EQUAL SIZE.               * 00175490
*                                                                     * 00176000
*       NOTE: THE ASA CONTROL 'CHARACTER PRINT-AND-SKIP-2' IS USED    * 00177000
*             FOR THE SUBMITTED LINES.                                * 00178000
*                                                                     * 00179000
*   7.  DISCONNECT THE COMMUNICATION PATH TO VSE/POWER.               * 00180000
*                                                                     * 00181000
*   8.  TERMINATE (LOG OFF FROM) THE VSE XPCC SUPPORT.                * 00182000
*                                                                     * 00183000
*   9.  TERMINATE PWRSASEX PROGRAM.                                   * 00183300
*                                                                     * 00183600
*********************************************************************** 00184000
*                                                                     * 00185000
*  THE FOLLOWING MACROS ARE REQUIRED:                                 * 00186000
*                                                                     * 00187000
*      SYSTEM MACROS:  XPCC                                           * 00188000
*                      XPCCB                                          * 00189000
*                      MAPXPCCB                                       * 00190000
*                      SETIME                                         * 00192000
*                      WAITM                                          * 00193000
*                      WAIT                                           * 00194000
*                                                                     * 00195000
*      VSE/POWER:      PWRSPL                                         * 00196000
*                                                                     * 00197000
*                                                                     * 00198000
*      AN INLINE MACRO (AVAILABLE WITH THE EXAMPLE).  IT IS USED FOR  * 00199000
*      DISPLAYING MESSAGES ON THE CONSOLE.  THE MACRO CALLS ARE IN    * 00200000
*      THE FORMAT:                                                    * 00201000
*                                                                     * 00202000
*          DPLAY MESSAGE-LABEL,LENGTH                                 * 00203000
*          DPLAY (REG1),(REG2)                                        * 00204000
*                                                                     * 00205000
*  NOTE: LINES WITH THE @-SIGN AT THE END REPRESENT THE INTERFACE     * 00206000
*        TO VSE/POWER.                                                * 00207000
*        LINES WITHOUT THE @-SIGN AT THE END REPRESENT THE INTERFACE  * 00208000
*        TO THE SYSTEM'S XPCC SUPPORT (STEPS 1, 2, 7, 8 AND 9).       * 00209490
*                                                                     * 00210000
*  CHANGE ACTIVITY:                                                   * 00211000
*                                                                     * 00212000
*   DO NOT CONNECT TO VSE/POWER IF XPCC IDENT RC >= X'08'     @DY43262* 00213000
*   VSE/POWER 6.1.1 TURBO DISPATCHER SHIPMENT                 @DY44055* 00214290
*   GUIDANCE FOR FIXED-FORMAT DISPLAY AND DIRECT GET REQUEST  @DY45495* 00214580
*********************************************************************** 00215000
         EJECT                                                          00216000
         SPACE 2                                                        00217000
*        REGISTER USAGE                                                 00218000
         SPACE 1                                                        00219490
*        R0 - **** - WORK REGISTER                                      00220000
*        R1 - **** - WORK REGISTER, ALSO USED BY PWRSPL MACRO           00221000
*        R2 - **** - WORK REGISTER                                      00222000
*        R3 - **** - WORK REGISTER                                      00223000
*        R4 - **** - ADDR REG FOR CROSS PARTITION CONTROL BLOCK XPCCB   00224000
*        R5 - **** - ADDRESS REGISTER FOR USER DATA TO BE SENT          00225000
*        R6 - **** - ADDRESS REGISTER FOR RECEIVED USER DATA            00226000
*        R7 - **** - ADDRESS REGISTER FOR SPL DSECT                     00227000
*        R8 - **** - FIRST BASE REGISTER OF PWRSASEX                    00228000
*        R9 - **** - SECOND BASE REGISTER OF PWRSASEX                   00229000
*        RA - **** - WORK REGISTER                                      00230000
*        RB - **** - WORK REGISTER                                      00231000
*        RC - **** - WORK REGISTER                                      00232000
*        RD - **** - BRANCH AND LINK REGISTER FOR SENDR SUBROUTINE      00233000
*        RE - **** - BRANCH AND LINK REGISTER FOR DATDSPLY SUBROUTINE   00234000
*        RF - **** - MACRO CALL RETURN CODE REGISTER                    00235000
         EJECT                                                          00236000
SAMPIN   START 120                START OF THIS SAMPLE PROGRAM          00237000
         BALR  R8,0               GET START ADDRESS                     00238000
         USING *,R8,R9            ESTABLISH ADDRESSABILITY              00239000
```

```
         SPACE 1                                                        00240490
         LA    R9,4095(,R8)        LOAD SECOND BASE REGISTER WITH       00241000
         LA    R9,1(,R9)           CONTENTS OF FIRST + 4096             00242000
         SPACE 1                                                        00243490
         LA    R4,OWNXPCCB         GET ADDR OF CROSS PART. CONTROL BLK  00244000
         USING IJBXPCCB,R4         ESTABLISH ADDRESSABILITY FOR DSECT   00245000
         SPACE 2                                                        00246000
         LA    R5,IJBXSUSR         GET ADDR OF USER DATA TO BE SENT     00247000
         USING PXUUSER,R5          ESTABLISH ADDRESSABILITY FOR DSECT   00248000
         SPACE 2                                                        00249000
         LA    R6,IJBXRUSR         GET ADDR OF RECEIVED USER DATA       00250000
         USING PXPUSER,R6          ESTABLISH ADDRESSABILITY FOR DSECT   00251000
         SPACE 2                                                        00252000
         LA    R7,OWNSPL           GET ADDR OF SPL                      00253000
         USING OWNSPLDS,R7         ESTABLISH ADDRESSABILITY FOR DSECT   00254000
         SPACE 2                                                        00254300
         MVC   FAILCOPY,FAILMSG    PRESERVE EMPTY MESSAGE SKELETON      00254600
         EJECT                                                          00255000
*********************************************************************** 00256000
**                      S T E P :   1                                ** 00257490
**          >> IDENTIFY PWRSASEX AS VSE/AF XPCC USER <<              ** 00257980
** IF THE MACRO FAILS, THE PROGRAM DISPLAYS A MESSAGE AND TERMINATES ** 00258470
**  - WITHOUT A DUMP IF IT FAILED BECAUSE OF LACK OF STORAGE         ** 00259000
**  - WITH A DUMP OTHERWISE.                                         ** 00260490
*********************************************************************** 00261000
         SPACE 1                                                        00262000
IDENT    DS    0H                                                       00263000
         SPACE 1                                                        00264000
         XPCC  XPCCB=(R4),FUNC=IDENT   IDENTIFY 'PWRSASEX' TO AF-XPCC   00265000
         SPACE 1                                                        00266000
         CLM   RF,M1,EIGHTDC       RETURN CODE < X'08'?                 00267590
         BL    CONCT               ..YES, CONNECT TO VSE/POWER          00268180
         SPACE 1                                                        00269000
         MVC   FAILFUNC,=C'IDENTIFY'  INSERT FAILING FUNCTION INTO MSG  00270000
         BAL   RE,MSGRETC          INSERT XPCC RETURN CODE INTO MSG     00271000
         MVC   FAILLABL,=C'IDENT ' INSERT CODE LABEL FOR DIAGNOSTIC     00272490
         BAL   RE,MSGDSPLY         DISPLAY MESSAGE ON CONSOLE           00273000
         CLI   IJBXRETC,IJBXNSTO   DID IDENT FAIL DUE TO NO STORAGE ?   00274000
         BE    FINEND              ..YES, TERMINATE WITH EOJ MACRO      00275000
         B     FINDUMP             BRANCH TO TERMINATION WITH DUMP      00276000
         EJECT                                                          00277000
*********************************************************************** 00278000
**                      S T E P :   2                                ** 00278500
**      >> ESTABLISH THE XPCC CONNECTION TO VSE/POWER <<             ** 00279000
** IF THE MACRO FAILS, THE PROGRAM DISPLAYS A FAILURE MESSAGE AND    ** 00280000
** TERMINATES.  THE PROGRAM WAITS UP TO TWO MINUTES FOR THE CONNEC-  ** 00281490
** TION TO BE COMPLETED.                                             ** 00282000
** IF THE CONNECTION IS ESTABLISHED AS REQUESTED, THE PROGRAM DIS-   ** 00283000
** PLAYS A CONFIRMATION MESSAGE.                                     ** 00284000
*********************************************************************** 00285000
         SPACE 1                                                        00286000
CONCT    DS    0H                                                       00287000
         SPACE 1                                                        00288000
         XPCC  XPCCB=(R4),FUNC=CONNECT        CONNECT TO VSE/POWER      00289000
         SPACE 1                                                        00290000
         LTR   RF,RF               IS CONNECTION ALREADY AVAILABLE ?    00291000
         BZ    CONNOK              ..YES, BYPASS WAIT FOR CONNECTION    00292000
         SPACE 1                                                        00293000
         CLM   RF,M1,EIGHTDC        WAS RETURN CODE X'08' GIVEN BACK ?  00294000
         BL    WAITCECB            ..NO, MUST BE '04', SO WAIT FOR CECB 00295000
         CLI   IJBXRETC,IJBXQSCE   DID VSE/POWER GIVE XPCC TERMQSCE ?   00296000
         BE    TERMQSCE            ..YES, GO TO HANDLE THAT STATE       00297000
         MVC   FAILFUNC,=C'CONNECT '   INSERT FAILING FUNCTION INTO MSG 00298000
         BAL   RE,MSGRETC          INSERT XPCC RETURN CODE INTO MSG     00299000
         MVC   FAILLABL,=C'CONCT ' INSERT CODE LABEL FOR DIAGNOSTIC     00300490
         BAL   RE,MSGDSPLY         DISPLAY MESSAGE ON CONSOLE           00301000
         CLI   IJBXRETC,IJBXNSTO   DID CONNECT FAIL DUE TO NO STOR. ?   00302000
```

```
         BE    TERMN               ..YES, GO TO CLOSE XPCC INTERFACE    00303000
         SPACE 1                                                        00304000
         B     FINDUMP             GO TO TERMINATION WITH DUMP          00305000
         SPACE 1                                                        00306000
TERMQSCE DS    0H                                                       00307000
         DPLAY FAILM1,72           DISPLAY FAILURE MESSAGE              00308000
         SPACE 1                                                        00309000
         B     TERMN               GO TO CLOSE XPCC INTERFACE CORRECTLY 00310000
         SPACE 1                                                        00311000
WAITCECB DS    0H                  CONNECTION IS STILL 'PENDING'        00312000
         SETIME 120,INTECB         INSTALL WAIT INTERVAL OF TWO MIN.    00313000
         LA    R3,IJBXCECB         LOAD ADDRESS OF CONNECTION ECB       00314000
         ST    R3,LISTCECB         COMPLETE WAITLIST                    00315000
         WAITM WAITLIST            WAIT FOR CONNECTION OR 2 MIN. COMPL. 00316000
         TM    IJBXCECB+2,POSTBIT  CONNECTION COMPLETE?                 00317000
         BO    CONNOK              ..YES, CONTINUE AT CONNOK            00318490
         SPACE 1                                                        00319000
         DPLAY FAILM3,72           ISSUE MSG THAT TIME LIMIT EXCEEDED   00320000
         SPACE 1                                                        00321000
         B     DISCT               GO TO DISCONNECT AND TERMINATE       00322000
         SPACE 1                                                        00323000
CONNOK   DS    0H                  NOW, CONNECTION ECB IS POSTED        00324000
         DPLAY SUCCM1,72                                                00325000
         EJECT                                                          00326000
***********************************************************************@ 00327000
**                    S T E P :   3                                   *@ 00327500
**      >>          PUT-REQUEST TO RDR QUEUE          <<              *@ 00328000
**  THE JOB 'EXAMPLE' IS SUBMITTED TO THE VSE/POWER RDR QUEUE.        *@ 00329000
***********************************************************************@ 00330000
         SPACE 1                                                       @ 00331000
*        REGISTER USAGE FOR PUT-REQUEST TO RDR QUEUE                   @ 00332000
         SPACE 2                                                       @ 00333000
*        R3 -  *****  - WORK REGISTER                                  @ 00334000
*        RA - BUFPTR  - POINTER FOR THE SEND BUFFER                    @ 00335000
*        RB - DATAPTR - POINTER FOR THE INPUT CARDS                    @ 00336000
*        RC -  *****  - TEMPORARY ADDR. REG FOR SPL DSECT              @ 00337000
         SPACE 2                                                       @ 00338000
*   THE GENERATED SPL (OWNSPL) IS UPDATED INDICATING A PUT OPEN        @ 00339000
*   REQUEST AND IS THEN SENT TO VSE/POWER.                             @ 00340490
         SPACE 2                                                       @ 00341000
PUTA1    DS    0H                                                      @ 00342000
         SPACE 1                                                       @ 00342100
         DPLAY SUCCM1A,72                                                00342200
         SPACE 1                                                       @ 00342300
*        THE SPL IS UPDATED FOR A 'PUT-OPEN JOB' REQUEST, SPECIFYING   @ 00342400
*         - THE MANDATORY FIELDS 'QUEUE, USERID'                       @ 00342500
*         FOR DETAILS ON MANDAT./OPT. FIELDS SEE PWRSPL REQ=PUT (JOB).@ 00342600
*         NOTE: THE JOB ATTRIBUTES WILL BE EXTRACTED FROM THE JECL     @ 00342700
*               JOB STATEMENT SUBMITTED LATER WITH THE JOB DATA.       @ 00342800
         SPACE 1                                                       @ 00342900
         PWRSPL TYPE=UPD,SPL=OWNSPL,REQ=PUT,QUEUE=RDR                  @ 00343000
         SPACE 2                                                       @ 00344000
         MVI   PXUBTYP,PXUBTSPL    INDICATE BUFFER TYPE = SPL          @ 00345000
         MVI   PXUACT1,0           CLEAR ALL OTHER BYTES IN PXUUSER,   @ 00346000
         MVI   PXUSIGNL,0          WHICH MAY BE CHANGED BY THE USER    @ 00347000
         SPACE 1                                                       @ 00348000
*        THE SPL IS DIRECTLY USED AS XPCC SEND BUFFER                  @ 00349000
         SPACE 1                                                       @ 00350000
         STCM  R7,M7,IJBXADR       INSERT SPL ADDRESS AS BUFFER ADDR.  @ 00351000
         LA    R3,SPLGLEN          LOAD LENGTH OF SPL                  @ 00352000
         ST    R3,IJBXBLN          INSERT BUFFER LENGTH INTO XPCCB     @ 00353000
         SPACE 1                                                       @ 00354000
         MVC   FAILLABL,=C'PUTA1 ' INSERT CODE LABEL FOR DIAGNOSTIC    @ 00355490
         BAL   RD,SENDR            GO TO SENDR ROUTINE                 @ 00356000
         CLI   PXPRETCD,PXPRCOK    WAS VSE/POWER RETURN CODE ZERO?     @ 00357000
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE    @ 00358000
*        THE VERIFICATION SPL RETURNED BY VSE/POWER FOR A PUT-OPEN     @ 00359190
```

```
*       REQUEST IS IGNORED, THE VERIFICATION SPL RETURNED LATER FOR  @ 00359380
*       A PUT-CLOSE REQUEST MAY BE OF MORE INTEREST.                 @ 00359570
        SPACE 2                                                      @ 00360000
*       FOR THE SUBSEQUENT 'PUT-SPOOL' REQUESTS THE PXU-USER FIELD   @ 00360300
*       SETTINGS ARE ESTABLISHED, AND                                @ 00360600
*       THE SEND BUFFER IS FILLED WITH INPUT CARDS (EACH CARD        @ 00361000
*       PRECEDED BY A RECORD PREFIX) UNTIL NO MORE CARD FITS.        @ 00362000
*       THE BUFFER IS THEN PASSED TO VSE/POWER IN THE ACTUALLY       @ 00363000
*       USED LENGTH.                                                 @ 00364000
        SPACE 1                                                      @ 00365000
        MVI   PXUBTYP,PXUBTNDB    BUFFER TYPE = NORMAL DATA BUFFER   @ 00366000
        MVI   PXUACT1,0           CLEAR ACTION BYTE                  @ 00367000
        SPACE 1                                                      @ 00368000
        LA    BUFPTR,SENDBUF      GET ADDRESS OF SEND BUFFER         @ 00369000
        STCM  BUFPTR,M7,IJBXADR   INSERT BUFFER ADDRESS INTO XPCCB   @ 00370000
        LA    DATAPTR,JECL1       GET ADDR OF FIRST INPUT CARD, ...  @ 00371290
*                                 USUALLY THE * $$ JOB STATEMENT     @ 00371580
        SPACE 1                                                      @ 00372000
FILLBUF DS    0H                                                     @ 00373000
        CLC   ENDIND,0(DATAPTR)   END OF FILE REACHED?             @ 0 00374490
        BE    PUTA3               YES, GO TO SEND FINAL BUFFER       @ 00375000
        CL    BUFPTR,LASTPREC     ENOUGH SPACE FOR ONE MORE RECORD?  @ 00376000
        BH    PUTA2               NO, GO TO SEND NORMAL BUFFER       @ 00377000
        USING RECPRFIX,BUFPTR     GET DSECT FOR RECORD LAYOUT        @ 00378000
        XC    0(RECPRFXL,BUFPTR),0(BUFPTR)   CLEAR BYTES FOR PREFIX  @ 00379000
        MVI   RECTYPE,RECTNORM    INSERT REC. TYPE IN REC. PREFIX    @ 00380000
        LA    R3,L'DATACARD       LOAD LENGTH OF DATA CARD           @ 00381000
        STH   R3,RECLNGTH         INSERT LENGTH OF DATA CARD IN PREF.@ 00382000
        LA    BUFPTR,RECPRFXL(,BUFPTR)        SKIP PREFIX IN BUFFER  @ 00383000
        DROP  BUFPTR                                                 @ 00384000
        MVC   0(L'DATACARD,BUFPTR),0(DATAPTR) MOVE DATA INTO BUFFER  @ 00385000
        LA    BUFPTR,L'DATACARD(,BUFPTR)   POINT TO NEXT FREE B.SPACE@ 00386000
        LA    DATAPTR,L'DATACARD(,DATAPTR) POINT TO NEXT INPUT CARD  @ 00387000
        B     FILLBUF             TRY TO FILL IN NEXT INPUT CARD     @ 00388000
        SPACE 1                                                      @ 00389000
PUTA2   DS    0H                                                     @ 00390000
        LA    R3,SENDBUF          GET AGAIN START ADDR OF SEND BUFFER@ 00391000
        SR    BUFPTR,R3           CALC. ACTUALLY USED BUFFER LENGTH  @ 00392000
        ST    BUFPTR,IJBXBLN      INSERT ACTUAL BUF.LENGTH INTO XPCCB@ 00393000
        MVC   FAILLABL,=C'PUTA2 ' INSERT CODE LABEL FOR DIAGNOSTIC   @ 00394490
        BAL   RD,SENDR            GO TO SENDR ROUTINE                @ 00395000
        CLI   PXPRETCD,PXPRCOK    WAS VSE/POWER RETURN CODE ZERO?    @ 00396000
        BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @ 00397000
        LA    BUFPTR,SENDBUF      GET AGAIN ADDRESS OF SEND BUFFER   @ 00398000
        B     FILLBUF             GO TO FILL BUFFER AGAIN            @ 00399000
        SPACE 2                                                      @ 00400190
*       FOR THE SUBSEQUENT 'PUT-CLOSE' REQUEST THE PXU-USER FIELD    @ 00400380
*       IS SET UP WITH THE END-OF-DATA INDICATION, AND               @ 00400570
*       THE BUFFER BEING FILLED WHEN END OF FILE WAS DETECTED        @ 00401000
*       IS PASSED TO VSE/POWER AS FINAL BUFFER.                      @ 00402990
        SPACE 1                                                      @ 00404000
PUTA3   DS    0H                                                     @ 00405000
        SPACE 1                                                      @ 00406000
        MVI   PXUACT1,PXUATEOD    INDICATE END OF DATA               @ 00407000
        LA    R3,SENDBUF          GET AGAIN START ADDR OF SEND BUFFER@ 00408000
        SR    BUFPTR,R3           CALC. ACTUALLY USED BUFFER LENGTH  @ 00409000
        ST    BUFPTR,IJBXBLN      INSERT ACTUAL BUF.LENGTH INTO XPCCB@ 00410000
        MVC   FAILLABL,=C'PUTA3 ' INSERT CODE LABEL FOR DIAGNOSTIC   @ 00411490
        BAL   RD,SENDR            GO TO SENDR ROUTINE                @ 00412000
        CLI   PXPRETCD,PXPRCOK    WAS VSE/POWER RETURN CODE ZERO?    @ 00413000
        BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @ 00414000
        CLI   PXPFBKCD,PXP000K    WAS POWER FEEDBACKCODE ALSO ZERO?  @ 00415000
        BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @ 00416000
        SPACE 1                                                      @ 00417000
*  THE VERIFICATION SPL RETURNED BY VSE/POWER IS ANALYZED, AND       @ 00418490
*  JOBNAME AND JOBNUMBER ARE SAVED.                                  @ 00419000
*  IF MESSAGES ARE QUEUED, A 'RETURN MESSAGE' REQUEST IS SENT. SUB-    00420490
```

```
*   SEQUENTLY, THE DATDSPLY ROUTINE IS CALLED IN ORDER TO DISPLAY     @ 00421000
*   THE RETURNED MESSAGES.                                            @ 00422000
         SPACE 1                                                      @ 00423000
         LA    RC,REPLBUF          GET AD. OF REPLY AREA FOR SPL DSECT@ 00424000
         DROP  R7                                                     @ 00425000
         USING OWNSPLDS,RC         ESTABLISH ADDRESSABILITY FOR DSECT @ 00426000
         MVC   JOBNAME,SPLGJB      SAVE JOBNAME RETURNED BY VSE/POWER @ 00427000
         MVC   JOBNUM,SPLGJN       SAVE RETURNED BINARY JOBNUMBER     @ 00428490
         DROP  RC                                                     @ 00429000
         USING OWNSPLDS,R7         REESTABLISH ADDRESSABILITY FOR SPL @ 00430000
         SPACE 1                                                      @ 00431000
         TM    PXPINFO,PXPIMSG     ARE MESSAGES QUEUED?               @ 00432000
         BZ    CTL1                NO, CONTINUE WITH CONTROL REQUEST  @ 00433490
         SPACE 1                                                      @ 00434000
PUTA4    DS    0H                                                     @ 00435000
         XC    IJBXBLN,IJBXBLN     INDICATE ZERO BUFFER LENGTH        @ 00436000
         MVI   PXUBTYP,0           CLEAR BUFFER TYPE BYTE IN USER DATA@ 00437000
         MVI   PXUACT1,PXUATRMR    INDICATE RETURN MESSAGE REQUEST    @ 00438000
         MVC   FAILLABL,=C'PUTA4 ' INSERT CODE LABEL FOR DIAGNOSTIC   @ 00439490
         BAL   RD,SENDR            GO TO SENDR ROUTINE                @ 00440000
         CLI   PXPRETCD,PXPRCOK    WAS VSE/POWER RETURN CODE ZERO?    @ 00441000
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @ 00442000
         BAL   RE,DATDSPLY         YES, GO TO DISPLAY RETURNED MSG'S  @ 00443000
         EJECT                                                        @ 00444000
*********************************************************************@ 00445000
**                     S T E P :   4                                 *@ 00446090
**              >>    CONTROL REQUEST    <<                          *@ 00446180
**                                                                   *@ 00446270
** COMMANDS CAN BE SUBMITTED TO VSE/POWER IN                         *@ 00446360
**  1) FIXED FORMAT - SEE 'PWRSPL FUNC=ALTER³CANCEL³DELETE³...       *@ 00446450
**                                   ...DISPLAY³HOLD³RELEASE', OR    *@ 00446540
**  2) FREE FORMAT  - SEE 'PWRSPL FUNC=COMMAND' IN THE FORMAT AS AN  *@ 00446630
**                                     OPERATOR WOULD KEY IT - FOR   *@ 00446720
**                                     ALL ALLOWED COMMANDS ACCEPTED *@ 00446810
**                                     VIA THE SPOOL-ACCESS INTERFACE*@ 00446900
**                                                                   *@ 00446990
** DISPLAY COMMANDS CAN REQUEST MESSAGES TO BE RETURNED AS           *@ 00447080
**  A) FIXED FORMAT RECORDS - SEE 'PWRSPL OPT=FORMAT', TO BE         *@ 00447170
**                            PROCESSED ACC. TO DSECT 'PXFMDSCT'     *@ 00447260
**  B) FREE FORMAT MESSAGES - SEE 'PWRSPL OPT=RESET', TO BE          *@ 00447350
**                            PROCESSED AS CONSOLE DISPLAY MESSAGES  *@ 00447440
**                                                                   *@ 00447530
** FOR PDISPLAY, THE COMMAND FORMATS 1) AND 2) CAN BE COMBINED WITH  *@ 00447620
** ANY MESSAGE FORMAT A) OR B). IN THE FOLLOWING, PWRSASEX OFFERS    *@ 00447710
** TWO COMBINATIONS AT LABEL                                         *@ 00447800
**  - 'CTLA1'  - CMD FORMAT 1) WITH MSG FORMAT B). THIS IS STANDARD  *@ 00447890
**                            FLOW, LEADS INTO LABEL 'GETB1' REQUEST.*@ 00447980
**  - 'CTLAB1' - CMD FORMAT 2) WITH MSG FORMAT A). THIS FLOW MUST BE *@ 00448070
**                            SELECTED, LEADS INTO GETBB1 REQUEST.   *@ 00448160
*********************************************************************@ 00450000
         SPACE 1                                                      @ 00451030
CTL1     DS    0H                                                     @ 00451060
         SPACE 1                                                      @ 00451090
         DPLAY SUCCM1B,72                                               00451120
         SPACE 1                                                      @ 00451150
         B     CTLA1               TAKE STANDARD FLOW, OR SELECT ...  @ 00451180
*        B     CTLAB1              ... ALTERNATIVE FLOW BY YOUR OWN   @ 00451210
         SPACE 1                                                      @ 00451240
         SPACE 1                                                      @ 00451270
*********************************************************************@ 00451300
**            >>  C T L  S T A N D A R D  F L O W  <<                *@ 00451330
** A FIXED FORMAT PDISPLAY COMMAND (FOR FREE FORMAT MESSAGES) IS     *@ 00451360
** SUBMITTED IN ORDER TO LOCATE THE OUTPUT OF JOB 'EXAMPLE' IN THE   *@ 00451390
** LST QUEUE (CLASS=S, ACC. TO * $$ LST) AND PRESENT THE LIST QUEUE  *@ 00451420
** DISPLAY LINE ON THE CONSOLE.                                      *@ 00451450
** NOTE: THE FREE FORMAT DISPLAY LINE(S) IS PRECEDED BY THE 'QUEUE   *@ 00451480
**       HEADER' LINE AS WITH NORMAL OPERATOR DISPLAY.               *@ 00451510
```

```
** NOTE: THE FIRST LST OUTPUT OF A JOB HAS ALWAYS THE SAME JOBNUMBER *@ 00451540
**       AS THE PARENT JOB. ITS JOBNUMBER HAS BEEN SAVED IN BINARY   *@ 00451570
**       FORMAT. IT IS NEEDED NOW IN THE SAME FORMAT.                *@ 00451600
*********************************************************************@ 00451630
         SPACE 1                                                     @ 00451660
*        REGISTER USAGE FOR CTL-REQUEST                              @ 00452000
         SPACE 1                                                     @ 00453490
*        RA - *****  - COUNTER FOR NUMBER OF WAIT INTERVALS          @ 00454000
         SPACE 1                                                     @ 00455190
*        THE SPL IS UPDATED FOR A DISPLAY-CTL REQUEST, SPECIFYING    @ 00455380
*         - THE MANDATORY SELECTION FIELDS 'QUEUE, JOBNAME, USERID'  @ 00455570
*         - PLUS OPTIONAL SELECTION FIELDS 'CLASS, JOBNUMBER'        @ 00455760
*         - PLUS RESETTING (FOR SAFETY) THE OPTIONAL 'OPT=..'.       @ 00455950
*         FOR DETAILS ON MANDATORY/OPTIONAL SEE PWRSPL REQ=CTL.      @ 00456140
         SPACE 1                                                     @ 00457000
CTLA1    DS    0H                                                    @ 00458000
         PWRSPL TYPE=UPD,SPL=OWNSPL,QUEUE=LST,REQ=CTL,CLASS=S,        *00459000
               JOBN=JOBNAME,JNUM=JOBNUM,FUNC=DISPLAY,OPT=RESET       @ 00460890
         SPACE 1                                                     @ 00461840
         LA    RA,12               PREPARE COUNTER FOR WAIT INTERVALS @ 00462000
         MVI   PXUBTYP,PXUBTSPL    INDICATE BUFFER TYPE = SPL        @ 00463000
         MVI   PXUACT1,0           CLEAR ACTION BYTE                 @ 00464000
         SPACE 1                                                     @ 00465000
*        THE UPDATED SPL IS DIRECTLY USED AS XPCC BUFFER.            @ 00466000
         SPACE 1                                                     @ 00467000
         STCM  R7,M7,IJBXADR       INSERT SPL ADDRESS AS BUFFER ADDR. @ 00468000
         SPACE 1                                                     @ 00469000
         LA    R3,SPLGLEN          LOAD LENGTH OF SPL                @ 00471990
         ST    R3,IJBXBLN          INSERT BUFFER LENGTH INTO XPCCB   @ 00475000
         SPACE 1                                                     @ 00476000
         MVC   FAILLABL,=C'CTLA2 ' INSERT CODE LABEL FOR DIAGNOSTIC  @ 00477490
CTLA2    DS    0H                                                    @ 00478000
         BAL   RD,SENDR            GO TO SENDR ROUTINE               @ 00479000
         SPACE 1                                                     @ 00482000
*    THE PROGRAM TESTS THE VSE/POWER RC/FBKCD TO SEE IF THE OUTPUT OF  @ 00483000
*    THE JOB 'EXAMPLE' COULD BE LOCATED.                             @ 00484000
*    IF THIS OUTPUT COULD NOT YET BE LOCATED, THE PROGRAM REPEATS THE  @ 00485000
*    CTL REQUEST EVERY 10 SECONDS IN ORDER TO WAIT FOR REQUEST       @ 00486690
*    COMPLETION.  HOWEVER PWRSASEX DISCONNECTS AFTER 12 UNSUCCESSFUL  @ 00487380
*    ATTEMPTS.                                                       @ 00488070
*    ANY OTHER RC/FBKCD COMBINATION SHOULD NOT OCCUR AND INDICATES A  @ 00489000
*    FAILURE OF THE REQUEST.                                         @ 00490000
         SPACE 1                                                     @ 00491000
         CLI   PXPRETCD,PXPRCOK    WAS VSE/POWER RETURN CODE ZERO?   @ 00491300
         BE    CTLA3               YES, CONTINUE WITH MSG DISPLAY    @ 00491600
         CLI   PXPRETCD,PXPRCOKF   WAS VSE/POWER RETURN CODE X'04'   @ 00492000
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE  @ 00493000
         CLI   PXPFBKCD,PXP04DNF   WAS OUTPUT NOT FOUND ? (PARENT ..  @ 00494290
*                                  .. JOB NOT YET COMPLETED)        )@ 00494580
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE  @ 00495000
         SPACE 1                                                     @ 00496000
         SETIME 10,INTECB          INSTALL WAIT INTERVAL OF 10 SEC.  @ 00497000
         WAIT   INTECB             WAIT                              @ 00498000
         BCT   RA,CTLA2            LOOP (MAX. 12 TIMES)              @ 00499000
         SPACE 1                                                     @ 00500000
         DPLAY FAILM4,72           DISPLAY FAILURE MESSAGE           @ 00501000
         SPACE 1                                                     @ 00502000
         B     DISCT               DISCONN AND TERMIN XPCC LINK, EOJ  @ 00503000
         SPACE 1                                                     @ 00504000
CTLA3    DS    0H                                                    @ 00505000
         SPACE 1                                                     @ 00506090
         DPLAY SUCCM1C,72                                              00506180
         SPACE 1                                                     @ 00506270
         BAL   RE,DATDSPLY         GO TO DISPLAY THE MESSAGE RETURNED @ 00506360
*                                  ...  IN 'CONSOLE DISPLAY' FORMAT  @ 00506450
         B     GETB1               GO FOR NORMAL GET-OPEN REQUEST    @ 00506540
         EJECT                                                       @ 00507000
```

```
         SPACE 1                                                       @ 00507006
***********************************************************************@ 00507012
**            >>  C T L  A L T E R N A T I V E  <<                    *@ 00507018
** A FREE FORMAT PDISPLAY COMMAND (FOR FIXED FORMAT MESSAGE) IS       *@ 00507024
** SUBMITTED IN ORDER TO LOCATE THE OUTPUT OF JOB 'EXAMPLE' IN THE    *@ 00507030
** LST QUEUE (CLASS=S, ACC. TO * $$ LST). SELECTED FIELDS OF THE      *@ 00507036
** FIXED FORMAT MESSAGE ARE PASSED TO THE ARTIFICIALLY BUILT '1QSAS'  *@ 00507042
** MESSAGE, WHICH IS THEN DISPLAYED ON THE CONSOLE.                   *@ 00507048
** NOTE: FIXED FORMAT MESSAGES ARE NOT PRECEDED BY A 'QUEUE HEADER'   *@ 00507054
**       LINE; QUEUE TYPE CAN BE DERIVED FROM PXFMQUID & -FLG1/-FLG3.*@ 00507060
** NOTE: THE FIRST LST OUTPUT OF A JOB HAS ALWAYS THE SAME JOBNUMBER *@ 00507066
**       AS THE PARENT JOB ITSELF. THE SAVED BINARY JOBNUMBER HAS     *@ 00507072
**       TO BE CONVERTED TO DECIMAL FOR USE IN THE PDISPLAY COMMAND. *@ 00507078
***********************************************************************@ 00507084
         SPACE 1                                                       @ 00507090
*        REGISTER USAGE FOR CTL-REQUEST                                @ 00507096
         SPACE 1                                                       @ 00507102
*        RA - *****  - COUNTER FOR NUMBER OF WAIT INTERVALS            @ 00507108
         SPACE 1                                                       @ 00507114
*        THE SPL IS UPDATED FOR A COMMON-CTL REQUEST, SPECIFYING       @ 00507120
*         - THE MANDATORY SELECTION FIELD 'USERID'                     @ 00507126
*         - PLUS SELECTION CRITERIA IN A FREE FORMAT COMMAND           @ 00507132
*         - PLUS THE OPTIONAL SELECTION CRITERION 'OPT=FORMAT'.        @ 00507138
*         FOR DETAILS ON MANDATORY/OPTIONAL SEE PWRSPL REQ=CTL.        @ 00507144
         SPACE 1                                                       @ 00507150
CTLAB1   DS    0H                                                      @ 00507156
         PWRSPL TYPE=UPD,SPL=OWNSPL,REQ=CTL,FUNC=COMMAND,               *00507162
               OPT=FORMAT                                              @ 00507168
         SPACE 1                                                       @ 00507174
*                                                                      @ 00507180
*        CONSTRUCT: 'PDISPLAY LST,<JOBNAME>,<JOBNUMBER>,CCLASS=S '     @ 00507186
*                                                                      @ 00507192
         SPACE 1                                                       @ 00507198
*        FEED JOBNAME TO FREE FORMAT COMMAND SKELETON                  @ 00507204
         SPACE 1                                                       @ 00507210
         MVC   CMDBODY(L'JOBNAME),JOBNAME  PLUG SAVED NAME INTO CMD    @ 00507216
         LA    R3,CMDBODY           POINT TO START OF JOBNAME          @ 00507222
CTLAB1A  DS    0H                                                      @ 00507228
         CLI   0(R3),C' '           FIRST TRAILING BLANK FOUND ?       @ 00507234
         BE    CTLAB1C              YES, GO TO PROVIDE 'COMMA'         @ 00507240
         LA    R3,1(R3)             PROCEED TO NEXT NAME BYTE          @ 00507246
         B     CTLAB1A              GO AND CHECK FOR BLANK             @ 00507252
CTLAB1C  DS    0H                                                      @ 00507258
         MVI   0(R3),C','           PROVIDE 'COMMA' AFTER JOBNAME      @ 00507264
         LA    R3,1(R3)             POINT TO BEGIN OF JOBNUMBER        @ 00507270
         SPACE 1                                                       @ 00507276
*        FEED DECIMAL JOBNUMBER TO FREE FORMAT COMMAND                 @ 00507282
         SPACE 1                                                       @ 00507288
         SR    R1,R1                CLEAR REGISTER                     @ 00507294
         ICM   R1,3,JOBNUM          PICK UP BINARY JOB NUMBER          @ 00507300
         CVD   R1,HELP8             CONVERT TO PACKED DECIMAL          @ 00507306
         UNPK  HELP5,HELP8+5(3)     UNPACK 3 DIGITS                    @ 00507312
         OI    HELP5+4,X'F0'        CHANGE X'C.' TO PRINTABLE X'F.'    @ 00507318
         MVC   0(5,R3),HELP5        PLUG 5 DIGIT JOBNUMBER WITH ...    @ 00507324
         LA    R3,5(R3)             POINT BEHIND JOBNUMBER             @ 00507330
         MVI   0(R3),C','           PROVIDE 'COMMA' AFTER JOBNAME      @ 00507336
         LA    R3,1(R3)             POINT TO BEGIN OF C-SELECTION FLD. @ 00507342
         SPACE 1                                                       @ 00507348
*        TERMINATE COMMAND BY 'CCLASS=S ', PASS COMMAND TO PWRSPL      @ 00507354
         SPACE 1                                                       @ 00507360
         MVC   0(L'CMDCLAS,R3),CMDCLAS  PASS CLASS SELECTION OPERAND   @ 00507366
         MVC   SPLCFLD,JOBCMD       PLUG FREE FORMAT CMD INTO PWRSPL   @ 00507372
         SPACE 1                                                       @ 00507378
         LA    RA,12                PREPARE COUNTER FOR WAIT INTERVALS @ 00507384
         MVI   PXUBTYP,PXUBTSPL     INDICATE BUFFER TYPE = SPL         @ 00507390
         MVI   PXUACT1,0            CLEAR ACTION BYTE                  @ 00507396
         SPACE 1                                                       @ 00507402
```

```
*       THE UPDATED SPL IS DIRECTLY USED AS XPCC BUFFER.           @ 00507408
        SPACE 1                                                    @ 00507414
        STCM  R7,M7,IJBXADR      INSERT SPL ADDRESS AS BUFFER ADDR. @ 00507420
        SPACE 1                                                    @ 00507426
        LA    R3,SPLGLEN         LOAD LENGTH OF SPL                @ 00507432
        ST    R3,IJBXBLN         INSERT BUFFER LENGTH INTO XPCCB   @ 00507438
        SPACE 1                                                    @ 00507444
        MVC   FAILLABL,=C'CTLAB2' INSERT CODE LABEL FOR DIAGNOSTIC  @ 00507450
CTLAB2  DS    0H                                                   @ 00507456
        BAL   RD,SENDR           GO TO SENDR ROUTINE               @ 00507462
        SPACE 1                                                    @ 00507468
*   THE PROGRAM TESTS THE VSE/POWER RC/FBKCD TO SEE IF THE OUTPUT OF  @ 00507474
*   THE JOB 'EXAMPLE' COULD BE LOCATED.                            @ 00507480
*   IF THIS OUTPUT COULD NOT YET BE LOCATED, THE PROGRAM REPEATS THE  @ 00507486
*   CTL REQUEST EVERY 10 SECONDS IN ORDER TO WAIT FOR REQUEST COMPLE- @ 00507492
*   TION.  HOWEVER PWRSASEX DISCONNECTS AFTER 12 UNSUCCESSFUL AT-     @ 00507498
*   TEMPTS.                                                        @ 00507504
*   ANY OTHER RC/FBKCD COMBINATION SHOULD NOT OCCUR AND INDICATES A   @ 00507510
*   FAILURE OF THE REQUEST.                                        @ 00507516
        SPACE 1                                                    @ 00507522
        CLI   PXPRETCD,PXPRCOK    WAS VSE/POWER RETURN CODE ZERO?   @ 00507528
        BE    CTLAB3             YES, CONTINUE WITH MSG DISPLAY    @ 00507534
        CLI   PXPRETCD,PXPRCOKF   WAS VSE/POWER RETURN CODE X'04'   @ 00507540
        BNE   REQFAIL            NO, GO TO HANDLE REQUEST FAILURE  @ 00507546
        CLI   PXPFBKCD,PXP04DNF   WAS OUTPUT NOT FOUND ? (PARENT ..  @ 00507552
*                                .. JOB NOT YET COMPLETED)        )@ 00507558
        BNE   REQFAIL            NO, GO TO HANDLE REQUEST FAILURE  @ 00507564
        SPACE 1                                                    @ 00507570
        SETIME 10,INTECB         INSTALL WAIT INTERVAL OF 10 SEC.  @ 00507576
        WAIT   INTECB            WAIT                              @ 00507582
        BCT   RA,CTLAB2          LOOP (MAX. 12 TIMES)              @ 00507588
        SPACE 1                                                    @ 00507594
        DPLAY FAILM4,72          DISPLAY FAILURE MESSAGE           @ 00507600
        SPACE 1                                                    @ 00507606
        B     DISCT              DISCONN AND TERMIN XPCC LINK, EOJ  @ 00507612
        SPACE 1                                                    @ 00507618
CTLAB3  DS    0H                                                   @ 00507624
        SPACE 1                                                    @ 00507630
        DPLAY SUCCM1C,72                                             00507636
        SPACE 1                                                    @ 00507642
        BAL   RE,DATDSPLY        GO TO INTERPRET THE FIXED FORMAT  @ 00507648
*                                MSG (ONLY ONE MESSAGE RECORD IS   @ 00507654
*                                EXPECTED) VIA PXFMDSCT, DISPLAY   @ 00507660
*                                '1QSAS', SAVE INTERNAL Q-ENTRY    @ 00507666
*                                NUMBER FOR THE SUBSEQUENT 'DIRECT' @ 00507672
*                                GET REQUEST.                      @ 00507678
        B     GETBB1             GO FOR 'DIRECT' GET-OPEN REQUEST  @ 00507684
        EJECT                                                      @ 00507690
*******************************************************************@ 00507696
**                  S T E P :   5                                 *@ 00507702
**           >>   GET REQUEST FROM LST QUEUE    <<                *@ 00507708
**                                                                *@ 00507714
** GET SERVICE REQUESTS CAN BE SUBMITTED TO VSE/POWER AS          *@ 00507720
**  1) 'GET FOR UPDATE' - ALLOWING ACCESS TO DISPATCHABLE ENTRIES, *@ 00507726
**                       AND TERMINATE GET BY 'CLOSE' (MAY DELETE  *@ 00507732
**                       ENTRY) OR BY 'QUIT' (PRESERVES ENTRY)     *@ 00507738
**  2) 'GET FOR BROWSE' - SEE PWRSPL MODE=BROWSE (SPLGFB1) FOR      *@ 00507744
**                       ACCESSING ENTRIES INDEPENDENT OF THEIR    *@ 00507750
**                       DISPOSITION, BUT TERMINATE GET REQUEST    *@ 00507756
**                       BY 'QUIT' ONLY.                           *@ 00507762
** BOTH GET REQUEST TYPES CAN BE INITIATED IN TWO FLAVOURS AS       *@ 00507768
**  A) NORMAL GET - WITH MANDATORY 'QUEUE, JOBNAME, CLASS, AND      *@ 00507774
**                       FROM/TO USERID ' AS PWRSPL SEARCH FIELDS  *@ 00507780
**  B) DIRECT GET - WITH SAME MANDATORY SEARCH FIELDS PLUS FIELD    *@ 00507786
**                       SPLXQNUM, SPECIFYING THE INTERNAL QUEUE   *@ 00507792
**                       ENTRY NUMBER - PROVIDED IT IS KNOWN BEFORE.*@ 00507798
**                       SEE ALSO "DIRECT QUEUE ENTRY GET ACCESS..."*@ 00507804
```

```
**                       IN THIS MANUAL FOR ADVANTAGES OF 'DIRECT'. *@ 00507810
**                                                                  *@ 00507816
** GET REQUEST TYPE 1) AND 2) CAN BE INITIATED WITH ANY A) OR B)    *@ 00507822
** SELECTION FLAVOURS. IN THE FOLLOWING, PWRSASEX OFFERS TWO        *@ 00507828
** COMBINATIONS AT LABEL                                            *@ 00507834
**  - 'GETB1'  - GET FOR UPDATE 1) WITH FLAVOUR A) 'NORMAL'. THIS IS *@ 00507840
**                           THE STANDARD CONTROL FLOW.             *@ 00507846
**  - 'GETBB1' - GET FOR BROWSE 2) WITH FLAVOUR B) 'DIRECT'. THIS   *@ 00507852
**                           FLOW IS SELECTED WHEN ENABLING 'CTLAB1'*@ 00507858
**                                                                  *@ 00507864
** N O T E :  REFER TO "SCOPE OF GET/CTL(NOT DISPLAY) ACCESS TO QUEUE*@ 00507870
**            ENTRIES" IN THIS MANUAL FIRST:                        *@ 00507876
**            THE SUBSEQUENT GET REQ. WITH INHERITED USERID=SASUSER1 *@ 00507882
**            IS ALLOWED TO ACCESS OUTPUT ENTRY 'EXAMPLE', BECAUSE  *@ 00507888
**            PARENT JOB WAS SUBMITTED BY 'PWRSPL USERID=SASUSER1', *@ 00507894
**            WHICH IS PROPAGATED TO ITS OUTPUT - HAVING ALSO THE   *@ 00507900
**            FROM/TO=SASUSER1 ATTRIBUTE.                           *@ 00507906
**            WHEN ACCESSING QUEUE ENTRIES WITH FROM/TO USERID NOT  *@ 00507912
**            MATCHING TO YOUR PWRSPL SPECIFICATION, CONSIDER TO    *@ 00507918
**            GENERATE VSE/POWER WITH A 'MASTER PASSWORD'. WHEN     *@ 00507924
**            SUPPLYING THIS PASSWORD (LEFT BOUND, PADDED BLANK) IN *@ 00507930
**            FIELD 'SPLGPW', YOUR GET/CTL REQUEST IS ENTITLED FOR  *@ 00507936
**            UNLIMITED ACCESS TO ANY QUEUE ENTRY.                  *@ 00507942
*********************************************************************@ 00507948
         SPACE 2                                                     @ 00507954
*********************************************************************@ 00508000
**            >>  G E T  S T A N D A R D  F L O W  <<               *@ 00509790
**  THE 'GET FOR UPDATE' SERVICE WITH 'NORMAL' SELECTION SPECIFIED  *@ 00510580
**  IS USED TO RETRIEVE THE LST QUEUE ENTRY OF JOB 'EXAMPLE' AND TO *@ 00511370
**  DISPLAY ITS DATA ON THE CONSOLE. THEN THE GET-RESTART FUNCTION  *@ 00512160
**  IS USED TO DISPLAY THE SECOND HALF OF THE ENTRY AGAIN.          *@ 00512950
*********************************************************************@ 00514000
         SPACE 1                                                     @ 00515000
*        REGISTER USAGE FOR GET-REQUEST FROM LST QUEUE               @ 00516000
         SPACE 1                                                     @ 00517490
*        R3 - ****    - WORK REGISTER                                @ 00518000
*        RA - BUFPTR  - POINTER FOR THE SEND BUFFER                  @ 00519000
         SPACE 1                                                     @ 00520090
GETB1    DS    0H                                                    @ 00520180
         DPLAY SUCCM7,72           DISPLAY MESSAGE                   @ 00520270
         SPACE 1                                                     @ 00520360
*        THE SPL IS UPDATED FOR A 'GET-OPEN' REQUEST, SPECIFYING     @ 00520450
*         - THE MANDATORY SELECTION 'CLASS, JOBNAME, QUEUE, USERID'  @ 00520540
*         - PLUS OPTIONAL SELECTION FIELD 'JOBNUMBER'                @ 00520630
*         - PLUS RESETTING (FOR SAFETY) THE OPTIONAL 'OPT=..'.       @ 00520720
*         FOR DETAILS ON MANDATORY/OPTIONAL SEE PWRSPL REQ=GET.      @ 00520810
* NOTE:  ONLY PARAMETERS WHICH ARE DIFFERENT FROM THOSE USED IN THE  @ 00520900
*        PREVIOUS CTL-REQUEST ARE SPECIFIED IN THE UPDATE SPL.       @ 00522000
         SPACE 1                                                     @ 00523000
         PWRSPL TYPE=UPD,SPL=(R7),REQ=GET,OPT=RESET                  @ 00525990
         SPACE 1                                                     @ 00528000
         MVI   PXUBTYP,PXUBTSPL    INDICATE BUFFER TYPE = SPL        @ 00529000
         MVI   PXUACT1,0           CLEAR ACTION BYTE 1               @ 00530000
         SPACE 1                                                     @ 00531000
         STCM  R7,M7,IJBXADR       INSERT SPL ADDRESS AS BUFFER ADDR. @ 00532000
         LA    R3,SPLGLEN          LOAD LENGTH OF SPL                @ 00533490
         ST    R3,IJBXBLN          INSERT BUFFER LENGTH INTO XPCCB   @ 00534000
         SPACE 1                                                     @ 00535000
         MVC   FAILLABL,=C'GETB1 ' INSERT CODE LABEL FOR DIAGNOSTIC  @ 00536490
         BAL   RD,SENDR            GO TO SENDR ROUTINE               @ 00537000
         CLI   PXPRETCD,PXPRCOK    WAS VSE/POWER RETURN CODE ZERO?   @ 00538000
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE  @ 00539000
         B     GETB2               GO AND TAKE STANDARD FLOW, WHEN   @ 00540010
*                                  COMING FROM 'CTLA1' CONTROL REQ.  @ 00540020
         SPACE 2                                                     @ 00540030
*********************************************************************@ 00540040
**            >>  G E T  A L T E R N A T I V E  <<                   *@ 00540050
```

```
**  THE 'GET FOR BROWSE' SERVICE WITH 'DIRECT' SELECTION SPECIFIED   *@ 00540060
**  IS USED TO RETRIEVE THE LST QUEUE ENTRY OF JOB 'EXAMPLE' AND TO  *@ 00540070
**  DISPLAY ITS DATA ON THE CONSOLE. THEN THE GET-RESTART FUNCTION   *@ 00540080
**  IS USED TO DISPLAY THE SECOND HALF OF THE ENTRY AGAIN.           *@ 00540090
*********************************************************************@ 00540100
         SPACE 1                                                      @ 00540110
*        REGISTER USAGE FOR GET-REQUEST FROM LST QUEUE                @ 00540120
         SPACE 1                                                      @ 00540130
*        R3 - ****    - WORK REGISTER                                 @ 00540140
*        RA - BUFPTR  - POINTER FOR THE SEND BUFFER                   @ 00540150
         SPACE 1                                                      @ 00540160
GETBB1   DS    0H                                                     @ 00540170
         DPLAY SUCCM7,72           DISPLAY MESSAGE                    @ 00540180
         SPACE 1                                                      @ 00540190
*        THE SPL IS UPDATED FOR 'DIRECT GET-OPEN' REQUEST, SPECIFYING @ 00540200
*         - THE MANDATORY SELECTION 'CLASS, JOBNAME, QUEUE, USERID'   @ 00540210
*         - PLUS OPTIONAL SELECTION FIELD 'MODE=BROWSE'               @ 00540220
*         - PLUS RESETTING (FOR SAFETY) THE OPTIONAL 'OPT=..'.        @ 00540230
*         FOR DETAILS ON MANDATORY/OPTIONAL SEE PWRSPL REQ=GET.       @ 00540240
         SPACE 1                                                      @ 00540250
         PWRSPL TYPE=UPD,SPL=(R7),REQ=GET,OPT=RESET,MODE=BROWSE,       *00540260
               JOBN=JOBNAME,CLASS=S,QUEUE=LST,JNUM=ZERONUM            @ 00540270
         SPACE 1                                                      @ 00540280
*        SEE "DIRECT QUEUE ENTRY GET (ALSO CTL) ACCESS ..." IN THIS   @ 00540290
*        MANUAL FOR REQUIRED PWRSPL SPECIFICTIONS.                    @ 00540300
         SPACE 1                                                      @ 00540310
         MVC   SPLXQNUM,JOBQNUM    SPECIFY SAVED Q-ENTRY-# FOR DIRECT @ 00540320
*                                  GET ACCESS TO DISPLAYED Q-ENTRY,   @ 00540330
*                                  NO JOBNUMBER NEEDED FOR UNIQUENESS @ 00540340
         OI    SPLGOPT2,SPLGO2QN   INDICATE 'USE QUEUE ENTRY NUMBER'  @ 00540350
*                                  TO ENABLE 'DIRECT' GET REQUEST     @ 00540360
         MVI   PXUBTYP,PXUBTSPL    INDICATE BUFFER TYPE = SPL         @ 00540370
         MVI   PXUACT1,0           CLEAR ACTION BYTE 1                @ 00540380
         SPACE 1                                                      @ 00540390
         STCM  R7,M7,IJBXADR       INSERT SPL ADDRESS AS BUFFER ADDR. @ 00540400
         LA    R3,SPLGLEN          LOAD LENGTH OF SPL                 @ 00540410
         ST    R3,IJBXBLN          INSERT BUFFER LENGTH INTO XPCCB    @ 00540420
         SPACE 1                                                      @ 00540430
         MVC   FAILLABL,=C'GETBB1' INSERT CODE LABEL FOR DIAGNOSTIC   @ 00540440
         BAL   RD,SENDR            GO TO SENDR ROUTINE                @ 00540450
         NI    SPLGOPT2,X'FF'-SPLGO2QN   RESET 'USE INT. Q-ENTRY-#'   @ 00540460
         XC    SPLXQNUM,SPLXQNUM   RESET INTERNAL QUEUE ENTRY NUMBER  @ 00540470
         CLI   PXPRETCD,PXPRCOK    WAS VSE/POWER RETURN CODE ZERO?    @ 00540480
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE,  @ 00540490
*                                  DISPLAY ALSO PXPFBKC2 IF NOT FOUND @ 00540500
         B     GETB2               GO AND JOIN COMMON FLOW NOW        @ 00540510
         SPACE 1                                                      @ 00540520
         EJECT                                                        @ 00540530
*   THE VERIFICATION SPL RETURNED BY VSE/POWER, WHICH COULD BE CHECKED@ 00541000
*   FOR USEFUL INFORMATION (SUCH AS FORMSID), IS IGNORED BY PWRSASEX. @ 00542000
*                                                                     @ 00543390
*   FOR THE SUBSEQUENT 'GET-SPOOL-DATA' REQUEST, THE PXU-USER FIELD   @ 00543780
*   IS FLAGGED WITH A SEND-DATA REQUEST, AND A NULL BUFFER IS PASSED  @ 00544170
*   TO VSE/POWER.                                                     @ 00544560
         SPACE 1                                                      @ 00545000
GETB2    DS    0H                                                     @ 00546000
         XC    IJBXBLN,IJBXBLN     INDICATE ZERO BUFFER LENGTH        @ 00547000
         MVI   PXUBTYP,0           CLEAR BUFFER TYPE BYTE IN USER DATA@ 00548000
         MVI   PXUACT1,PXUATSDR    INDICATE SEND DATA REQUEST         @ 00549000
         MVC   FAILLABL,=C'GETB2 ' INSERT CODE LABEL FOR DIAGNOSTIC   @ 00550490
         BAL   RD,SENDR            GO TO SENDR ROUTINE                @ 00551000
         CLI   PXPRETCD,PXPRCOK    WAS VSE/POWER RETURN CODE ZERO?    @ 00552000
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @ 00553000
         MVI   GETFCT,C'G'         INDICATE: DATDSPLY IS CALLED BY GET@ 00554000
         BAL   RE,DATDSPLY         GO TO DISPLAY RETURNED DATA        @ 00555000
*                                  AND DO NOT RETURN UNTIL LAST DATA  @ 00556000
*                                  RECORD IS DISPLAYED                @ 00557000
```

```
         MVI   GETFCT,C' '         RESET INDICATION                    @ 00558000
         SPACE 1                                                       @ 00559000
         DPLAY SUCCM2,72           DISPLAY MSG TO INDICATE RESTART RQ.@ 00560000
         SPACE 2                                                       @ 00561590
*        A 'RESTART CONTROL RECORD' IS BUILT IN THE SEND BUFFER AND   @ 00562180
*        PASSED TO VSE/POWER. THE LOGICAL RECORD NUMBER - PREVIOUSLY  @ 00563000
*        SAVED BY THE DATDSPLY ROUTINE - IS USED AS RESTART POINT.    @ 00564490
         SPACE 1                                                       @ 00565000
GETB3    DS    0H                                                      @ 00566000
         SPACE 1                                                       @ 00567000
         MVI   PXUBTYP,PXUBTCTL    BUFFER TYPE = CONTROL RECORD        @ 00568000
         MVI   PXUACT1,0           CLEAR ACTION BYTE 1                 @ 00569000
         LA    BUFPTR,SENDBUF      GET ADDRESS OF SEND BUFFER          @ 00570000
         STCM  BUFPTR,M7,IJBXADR   INSERT BUFFER ADDRESS INTO XPCCB    @ 00571000
         SPACE 1                                                       @ 00572000
         USING PXRSDSCT,BUFPTR     GET DSECT FOR RESTART CONTROL REC. @ 00573000
         XC    0(PXRSLENG,BUFPTR),0(BUFPTR)   CLEAR RESTART CONTROL R.@ 00574000
         MVI   PXRSTYPE,PXRSTRST   INDICATE RECORD TYPE = RESTART CTL.@ 00575000
         MVC   PXRSRECN,PWRRECNO   INSERT PREVIOUSLY SAVED LOG. REC.# @ 00576000
         LA    R3,PXRSLENG         LOAD LENGTH OF RESTART CTL. REC.   @ 00577000
         STH   R3,PXRSRLEN         INSERT LENGTH INTO RESTART CTL. REC@ 00578000
         ST    R3,IJBXBLN          INSERT LENGTH INTO XPCCB            @ 00579000
         DROP  BUFPTR                                                  @ 00580000
         SPACE 1                                                       @ 00581000
         MVC   FAILLABL,=C'GETB3 ' INSERT CODE LABEL FOR DIAGNOSTIC   @ 00582490
         BAL   RD,SENDR            GO TO SENDR ROUTINE                 @ 00583000
         CLI   PXPRETCD,PXPRCOK    WAS VSE/POWER RETURN CODE ZERO?    @ 00584000
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @ 00585000
         MVI   GETFCT,C'G'         INDICATE: DATDSPLY IS CALLED BY GET@ 00586000
         BAL   RE,DATDSPLY         YES, GO TO DISPLAY RETURNED DATA   @ 00587000
*                                  AND DO NOT RETURN UNTIL LAST DATA  @ 00588000
*                                  RECORD IS DISPLAYED                 @ 00589000
         MVI   GETFCT,C' '         RESET INDICATION                    @ 00590000
         SPACE 1                                                       @ 00591000
*    FOR THE SUBSEQUENT 'GET-QUIT' REQUEST, THE PXU-USER FIELD IS      @ 00592490
*    FLAGGED WITH A 'QUIT' INDICATION, AND A NULL BUFFER IS PASSED     @ 00592980
*    TO VSE/POWER.                                                     @ 00593470
         SPACE 1                                                       @ 00594000
GQUIT    DS    0H                                                      @ 00595000
         XC    IJBXBLN,IJBXBLN     INSERT ZERO BUFFER LENGTH           @ 00596000
         MVI   PXUBTYP,0           CLEAR BUFFER TYPE BYTE IN USER DATA@ 00597000
         MVI   PXUACT1,PXUATABR    INDICATE QUIT REQUEST               @ 00598000
         SPACE 1                                                       @ 00599000
*        IF A CLOSE OR PURGE REQUEST IS DESIRED (ONLY IN CASE OF       @ 00600590
*        'NORMAL' GET), ONE OF THE FOLLOWING STATEMENTS MUST BE CODED @ 00601180
         SPACE 1                                                       @ 00602000
*GCLOSE  MVI   PXUACT1,PXUATRQS    REQUIRED SETTING FOR A CLOSE REQU. @ 00603000
*GPURGE  MVI   PXUACT1,PXUATPRG    REQUIRED SETTING FOR A PURGE REQU. @ 00604000
         SPACE 1                                                       @ 00605000
         MVC   FAILLABL,=C'GQUIT ' INSERT CODE LABEL FOR DIAGNOSTIC   @ 00606490
         BAL   RD,SENDR            GO TO SENDR ROUTINE                 @ 00607000
         CLI   PXPRETCD,PXPRCOK    WAS VSE/POWER RETURN CODE ZERO?    @ 00608000
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @ 00609000
         SPACE 2                                                       @ 00610000
         EJECT                                                         @ 00611000
**********************************************************************@ 00612000
**                    S T E P :   6                                   *@ 00612500
**          >>      PUT REQUEST TO LST QUEUE          <<              *@ 00613000
**   THE DATA CARDS OF THE EXAMPLE JOB ARE SUBMITTED TO THE VSE/POWER *@ 00614000
**   LST QUEUE AS 'EXAMPSEG'.  A SEGMENT REQUEST IS ISSUED            *@ 00615000
**   AFTER EACH SEVENTH RECORD.                                       *@ 00616000
**********************************************************************@ 00617000
         SPACE 1                                                       @ 00618000
*        REGISTER USAGE FOR PUT-REQUEST TO LST QUEUE                   @ 00619000
         SPACE 2                                                       @ 00620000
*        R2 - RECORDCT - RECORD COUNTER FOR SEGMENTATION IN LOOP       @ 00621000
*        RA - BUFPTR   - POINTER FOR THE SEND BUFFER                   @ 00622000
```

```
*       RB - DATAPTR  - POINTER FOR THE INPUT CARDS                 @ 00623000
        SPACE 2                                                     @ 00624000
PUTB1    DS    0H                                                   @ 00625990
        SPACE 1                                                     @ 00626980
        DPLAY SUCCM8,72           DISPLAY MESSAGE                   @ 00628000
        SPACE 1                                                     @ 00629000
*       THE SPL IS UPDATED FOR A 'PUT-OPEN OUTPUT' REQ., SPECIFYING  @ 00630090
*        - THE MANDATORY FIELDS 'QUEUE, JOBNAME, USERID', AND       @ 00630180
*        - THE OPTIONAL FIELDS 'CLASS, MODE, (OPT, PWD)' FOR UPD REQ.@ 00630270
*       FOR DETAILS ON MANDAT./OPT. FIELDS SEE PWRSPL REQ=PUT OUTPUT.@ 00630360
*       NOTE: ALL FURTHER OUTPUT ATTRIBUTES HAVE TO BE SET BY OWN    @ 00630450
*             CODE, TO FEED UPDATE SPL FIELDS ACC. TO "SUBMITTING    @ 00630540
*             OUTPUT DATA" IN THIS MANUAL.                          @ 00630630
        SPACE 1                                                     @ 00630720
        PWRSPL TYPE=UPD,REQ=PUT,SPL=OWNSPL,CLASS=Z,JOBN=JOBNLAB,      *00631000
              QUEUE=LST,MODE=RESET                                  @ 00632000
        SPACE 2                                                     @ 00633000
*   SET ADDITIONAL OUTPUT SPECIFIC FIELDS IN THE SPL BY OWN CODE     @ 00634490
        MVI   SPLDDP,DISP         INDICATE OUTPUT DISPOSITION       @ 00635000
        MVI   SPLONSEP,SEPPAGE    INDICATE OUTPUT SEPARATOR PAGES   @ 00636000
        MVI   SPLDPR,PRIOR        INDICATE OUTPUT PRIORITY          @ 00637000
        MVC   SPLOFORM,FORMS      INDICATE OUTPUT FORMS             @ 00638000
        MVI   SPLORCFM,SPLORASA   INDICATE ASA CC FOR OUTPUT        @ 00639000
        MVI   SPLDSID,C'N'        INDICATE 'NO' SPECIFIC TARGET SYSID@ 00639300
*                                 FIELD IS REQUIRED FOR SHARED SYST. @ 00639600
        SPACE 1                                                     @ 00640000
        MVI   PXUACT1,0           CLEAR ACTION BYTE 1 IN USER DATA  @ 00641000
        MVI   PXUBTYP,PXUBTSPL    INDICATE BUFFER TYPE = SPL        @ 00642000
        SPACE 1                                                     @ 00643000
        STCM  R7,M7,IJBXADR       INSERT SPL ADDRESS AS BUFFER ADDR. @ 00644000
        LA    R3,SPLGLEN          LOAD LENGTH OF SPL                @ 00645000
        ST    R3,IJBXBLN          INSERT BUFFER LENGTH INTO XPCCB   @ 00646000
        SPACE 1                                                     @ 00647000
        MVC   FAILLABL,=C'PUTB1 ' INSERT CODE LABEL FOR DIAGNOSTIC  @ 00648490
        BAL   RD,SENDR            GO TO SENDR ROUTINE               @ 00649000
        CLI   PXPRETCD,PXPRCOK    WAS VSE/POWER RETURN CODE ZERO?   @ 00650000
        BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE  @ 00651000
        SPACE 2                                                     @ 00652000
*   THE VERIFICATION SPL RETURNED BY VSE/POWER IS IGNORED.          @ 00653390
*                                                                   @ 00653780
*   FOR THE SUBSEQUENT 'PUT-SPOOL-DATA' REQUEST, THE PXU-USER FIELD  @ 00654170
*   SETTINGS ARE ESTABLISHED, AND THE SEND BUFFER IS FILLED WITH     @ 00654560
*   'OUTPUT LINE RECORDS' (EACH RECORD PRECEDED BY A RECORD PREFIX)  @ 00654950
*   UNTIL NO MORE RECORD FITS.                                      @ 00655340
*   THE BUFFER IS THEN PASSED TO VSE/POWER IN THE ACTUALLY USED      @ 00655730
*   LENGTH. THE OUTPUT IS ALWAYS SEGMENTED AFTER SEVEN RECORDS.      @ 00656120
        SPACE 1                                                     @ 00657000
PUTB2    DS    0H                                                   @ 00658000
        MVI   PXUBTYP,PXUBTNDB    BUFFER TYPE = NORMAL DATA BUFFER  @ 00659000
        MVI   PXUACT1,0           CLEAR ACTION BYTE 1 IN USER DATA  @ 00660000
        SPACE 1                                                     @ 00661000
        LA    BUFPTR,SENDBUF      GET ADDRESS OF SEND BUFFER        @ 00662000
        STCM  BUFPTR,M7,IJBXADR   INSERT BUFFER ADDRESS INTO XPCCB  @ 00663000
        LA    DATAPTR,DATACARD    GET ADDR OF FIRST INPUT CARD      @ 00664000
        SPACE 1                                                     @ 00665000
        LA    RECORDCT,NOOFRECS   INITIALIZE RECORD COUNTER         @ 00666000
FILLBUFO DS    0H                                                   @ 00667000
        CLC   JCL2(3),0(DATAPTR) END OF DATA REACHED?               @ 00668000
        BE    SDEOD               YES, GO TO SEND FINAL BUFFER      @ 00669000
        SPACE 1                                                     @ 00670000
        CL    BUFPTR,LASTPREC     ENOUGH SPACE FOR ONE MORE RECORD?  @ 00671000
        BH    SDNDB               NO, GO TO SEND NORMAL BUFFER      @ 00672000
        SPACE 1                                                     @ 00673000
        USING RECPRFIX,BUFPTR     GET DSECT FOR RECORD LAYOUT       @ 00674000
        XC    0(RECPRFXL,BUFPTR),0(BUFPTR)   CLEAR BYTES FOR PREFIX  @ 00675000
        MVI   RECTYPE,RECTNORM    INSERT REC. TYPE INTO REC. PREFIX  @ 00676000
        MVI   RECCCODE,C'-'       SET ASA CC IN REC. PREFIX TO SKIP2 @ 00677000
```

```
         LA    R3,NOOFRECS         MAX NUMBER OF RECORDS IN A SEGMENT @ 00678000
         CLR   RECORDCT,R3         FIRST RECORD OF SEGMENT?           @ 00679000
         BNE   LAB1                NO, CONTINUE AT LABEL LABL1        @ 00680000
         MVI   RECCCODE,C'1'       SET ASA CC IN REC.PREF. TO NXT.PAGE@ 00681000
         SPACE 1                                                      @ 00682000
LAB1     DS    0H                                                     @ 00683000
         LA    R3,L'DATACARD       LOAD LENGTH OF DATA                @ 00684000
         STH   R3,RECLNGTH         INSERT LENGTH OF DATACARD IN PREFIX@ 00685000
         LA    BUFPTR,RECPRFXL(,BUFPTR) SKIP PREFIX IN BUFFER         @ 00686000
         MVC   0(L'DATACARD,BUFPTR),0(DATAPTR) MOVE DATA IN BUFFER    @ 00687000
         LA    BUFPTR,L'DATACARD(,BUFPTR) POINT TO NEXT FREE BUFSPACE @ 00688000
         LA    DATAPTR,L'DATACARD(,DATAPTR) POINT TO NEXT INPUT CARD  @ 00689000
         BCT   RECORDCT,FILLBUFO   DO LOOP AND DECREMENT RECORDCOUNTER@ 00690000
         SPACE 1                                                      @ 00691000
         CLC   JCL2(3),0(DATAPTR) END OF DATA REACHED?                @ 00692000
         BE    SDEOD               YES, GO TO SEND FINAL BUFFER       @ 00693000
         MVI   PXUACT1,PXUATSGM    INDICATE OUTPUT SEGMENTATION       @ 00694000
SDNDB    DS    0H                                                     @ 00695000
         LA    R3,SENDBUF          GET AGAIN START ADDR. OF SEND BUFF @ 00696000
         SR    BUFPTR,R3           CALC. ACTUALLY USED BUFFER LENGTH  @ 00697000
         ST    BUFPTR,IJBXBLN      INSERT ACTUAL BUF.LENGTH INTO XPCCB@ 00698000
         MVC   FAILLABL,=C'SDNDB ' INSERT PART OF CODE LABEL FOR DIAGN@ 00699490
         BAL   RD,SENDR            GO TO SENDR ROUTINE                @ 00700000
         CLI   PXPRETCD,PXPRCOK    WAS VSE/POWER RETURN CODE ZERO?    @ 00701000
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @ 00702000
         LA    BUFPTR,SENDBUF      GET ADDRESS OF SEND BUFFER         @ 00703000
         LTR   RECORDCT,RECORDCT   IS RECORD COUNTER ZERO?            @ 00704000
         BNZ   KEEPRCT             NO, KEEP ACTUAL VALUE OF RECORDCT  @ 00705000
         LA    RECORDCT,NOOFRECS   INITIALIZE REC COUNTER AGAIN       @ 00706000
KEEPRCT  DS    0H                                                     @ 00707000
         MVI   PXUACT1,0           CLEAR ACTION BYTE 1 IN USER DATA   @ 00708000
         B     FILLBUFO            GOTO CHECK NEXT INPUT CARD         @ 00709000
         SPACE 2                                                      @ 00710000
SDEOD    DS    0H                                                     @ 00711000
         MVI   PXUACT1,0           CLEAR ACTION BYTE 1 IN USER DATA   @ 00712000
         MVI   PXUACT1,PXUATEOD    ACTION BYTE = END OF DATA          @ 00713000
         LA    R3,SENDBUF          GET AGAIN START ADDR. OF SEND BUFF @ 00714000
         SR    BUFPTR,R3           CALC. ACTUALLY USED BUFFER LENGTH  @ 00715000
         ST    BUFPTR,IJBXBLN      INSERT ACTUAL BUF.LENGTH INTO XPCCB@ 00716000
         L     R3,IJBXBLN          LOAD ACTUAL SEND BUFFER LENGTH     @ 00717000
         LTR   R3,R3               IS BUFFER LENGTH  ZERO?            @ 00718000
         BNZ   NOTNLB              NO, IND. NORMAL DATA BUFFER        @ 00719000
         MVI   PXUBTYP,0           CLEAR BUFFER TYPE IN USER DATA     @ 00720000
NOTNLB   DS    0H                                                     @ 00721000
         MVC   FAILLABL,=C'SDEOD ' INSERT PART OF CODE LABEL FOR DIAGN@ 00722490
         BAL   RD,SENDR            GO TO SENDR ROUTINE                @ 00723000
         CLI   PXPRETCD,PXPRCOK    WAS VSE?POWER RETURN CODE ZERO?    @ 00724000
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @ 00725000
         SPACE 2                                                      @ 00726000
*        THE EXTENDED SPL RETURNED BY VSE/POWER IS IGNORED.           @ 00727000
         SPACE 1                                                      @ 00728000
PUTB3    DS    0H                                                     @ 00729000
         TM    PXPINFO,PXPIMSG     MESSAGES QUEUED?                   @ 00730000
         BNO   DISPLAY             NO, GO TO DISPLAY INFO MESSAGES    @ 00731000
         MVC   FAILLABL,=C'PUTB3 ' INSERT CODE LABEL FOR DIAGNOSTIC   @ 00732490
         XC    IJBXBLN,IJBXBLN     INDICATE ZERO BUFFER LENGTH        @ 00733000
         MVI   PXUBTYP,0           CLEAR USER DATA IN XPCCB           @ 00734000
         MVI   PXUACT1,PXUATRMR    INDICATE RETURN QUEUED MESSAGES    @ 00735000
         BAL   RD,SENDR            GO TO SENDR ROUTINE                @ 00736000
         CLI   PXPRETCD,PXPRCOK    WAS VSE/POWER RETURN CODE ZERO?    @ 00737000
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE   @ 00738000
         BAL   RE,DATDSPLY         YES, GO TO DISPLAY RETURNED MSG'S  @ 00739000
         SPACE 1                                                      @ 00740000
DISPLAY  DS    0H                                                     @ 00741000
         DPLAY SUCCM3,72           DISPLAY MESSAGE                    @ 00742000
         DPLAY SUCCM4,72           DISPLAY MESSAGE                    @ 00743000
         DPLAY SUCCM6,72           DISPLAY MESSAGE                    @ 00744000
```

```
         SPACE 1                                                     @ 00745000
         B     DISCT              DISCONN AND TERMIN XPCC LINK, EOJ  @ 00746000
         EJECT                                                       @ 00747000
********************************************************************** 00834000
**                     S T E P :   7                                ** 00834500
**    >> DISCONNECT THE XPCC COMMUNICATION LINK TO VSE/POWER <<     ** 00835000
** IF THE MACRO FAILS, THE PROGRAM DISPLAYS A DIAGNOSTIC MESSAGE AND ** 00836000
** TERMINATES WITH A DUMP.                                          ** 00837000
********************************************************************** 00838000
         SPACE 1                                                       00839000
DISCT    DS    0H                                                      00840000
         XPCC  XPCCB=(R4),FUNC=DISCONN   DISCONNECT LINK TO VSE/POWER   00841000
         SPACE 1                                                       00842000
         LTR   RF,RF              WAS DISCONNECT SUCCESSFUL, RF='00' ? 00843000
         BZ    TERMN              ..YES CONTINUE WITH XPCC TERMINATION 00844000
         SPACE 1                                                       00845000
         MVC   FAILFUNC,=C'DISCONN '   INSERT FAILING FUNCTION         00846000
         BAL   RE,MSGRETC         INSERT XPCC RETURN CODE INTO MSG     00847000
         MVC   FAILLABL,=C'DISCT ' INSERT CODE LABEL FOR DIAGNOSTIC    00848490
         BAL   RE,MSGDSPLY        DISPLAY MESSAGE ON CONSOLE           00849000
         B     FINDUMP            GO TO TERMINATION WITH DUMP          00850000
         EJECT                                                         00851000
********************************************************************** 00852000
**                     S T E P :   8                                ** 00852500
**    >> TERMINATE INTERACTION WITH THE VSE/AF XPCC SUPPORT <<      ** 00853000
** IF THE MACRO FAILS, THE PROGRAM DISPLAYS A DIAGNOSTIC MESSAGE AND ** 00854000
** TERMINATES WITH A DUMP.                                          ** 00855000
********************************************************************** 00856000
         SPACE 1                                                       00857000
TERMN    DS    0H                                                      00858000
         XPCC XPCCB=(R4),FUNC=TERMIN   TERMINATE CROSS PART. INTERFACE  00859000
         LTR   RF,RF              DID WE GET A ZERO RET-CODE ?         00860000
         BZ    FINEND             ..YES, GO TO NORMAL EOJ MACRO        00861000
         SPACE 1                                                       00862000
         MVC   FAILFUNC,=C'TERMIN  '  INSERT FAILING FUNCTION INTO MSG  00863000
         BAL   RE,MSGRETC         INSERT XPCC RETURN CODE INTO MSG     00864000
         MVC   FAILLABL,=C'TERMN ' INSERT CODE LABEL FOR DIAGNOSTIC    00865490
         BAL   RE,MSGDSPLY        DISPLAY MESSAGE ON CONSOLE           00866000
         B     FINDUMP            GO TO TERMINATION WITH DUMP          00867000
         EJECT                                                         00868000
********************************************************************** 00869000
**                     S T E P :   9                                ** 00869500
**                  >> TERMINATE PWRSASEX  <<                       ** 00870000
********************************************************************** 00871000
         SPACE 1                                                       00872000
FINDUMP  DS    0H                 TERMINATION FORCED DUE TO ERROR      00873000
*        DUMP                     A PARTITION DUMP CAN BE FORCED IF    00874000
*                                 NECESSARY FOR DEBUG PURPOSES         00875000
         SPACE 1                                                       00876000
FINEND   DS    0H                 NORMAL TERMINATION                   00877000
         EOJ                      NORMAL END OF PWRSASEX PROGRAM       00878000
         EJECT                                                         00879000
********************************************************************** 00879008
**                     S U B R O U T I N E S                        ** 00879016
**                  >>   DATDSPLY ROUTINE      <<                   ** 00879024
**  THIS ROUTINE DISPLAYS THE FOLLOWING INFO RETURNED BY VSE/POWER:  ** 00879032
**   - MESSAGES IN 'OPERATOR' DISPLAY FORMAT (CTL OR PUT REQUEST)   ** 00879040
**   - MESSAGES IN 'FIXED FORMAT' (CTL OPT=FORMAT), CONVERTED TO THE ** 00879048
**     OWN CONSOLE MESSAGE '1QSAS'                                  ** 00879056
**   - DATA RECORDS OF RETRIEVED QUEUE ENTRIES (GET REQUEST)        ** 00879064
********************************************************************** 00879072
         SPACE 1                                                       00879080
*        REGISTER USAGE FOR DATDSPLY ROUTINE                           00879088
         SPACE 1                                                       00879096
*        RA - BUFPTR  - POINTER FOR THE REPLY BUFFER                   00879104
*        RC - BUFLN   - REG TO CALCULATE THE LENGTH OF THE DATA STILL   00879112
*                       TO BE DISPLAYED                                00879120
```

```
*       R0, R1, R2, R3 - WORK REGISTER
*
*       CALLED FROM: PUT REQUEST TO RDR QUEUE
*                    CTL REQUEST
*                    GET REQUEST
*                    PUT REQUEST TO LST QUEUE
*
*       EXIT TO CALLER IF ALL AVAILABLE MESSAGES/DATA ARE DISPLAYED
         SPACE 2
DATDSPLY DS    0H
         SR    R0,R0              SET R0 TO ZERO
         CLM   R0,M7,IJBXSLN      NO MORE DATA TO DISPLAY?
         BER   RE                 RETURN TO CALLER
         LA    BUFPTR,REPLBUF     POINT TO REPLY BUFFER
         SR    BUFLN,BUFLN        CLEAR REGISTER
         ICM   BUFLN,M7,IJBXSLN   GET LENGTH OF DATA TO BE DISPLAYED
         SPACE 1
*   PWRSASEX DISPLAYS, RECORD AFTER RECORD, THE DATA OR MESSAGES RE-
*   TURNED BY VSE/POWER.  THE RECORD PREFIX OF EACH DATA RECORD IS
*   ANALIZED BUT NOT DISPLAYED.
*   IF DATDSPLY IS CALLED TO DISPLAY PARTS OF A FIXED FORMAT MESSAGE
*   RECORD (ONLY ONE EXPECTED), THE RETURNED 'INTERNAL' QUEUE ENTRY
*   NUMBER OF OUTPUT 'EXAMPLE' IS SAVED FOR THE ALTERNATIVE FLOW OF THE
*   'DIRECT GET' REQUEST.
*   IF DATDSPLY IS CALLED BY THE GET FUNCTION, THE LOGICAL RECORD
*   NUMBER OF THE 12TH DATA CARD OF OUTPUT ENTRY 'EXAMPLE' IS SAVED.
*   PWRSASEX USES THIS NUMBER LATER AS A RESTART POINT.
         SPACE 1
DSPL0    DS    0H
         USING RECPRFIX,BUFPTR    GET DSECT OF RECORD LAYOUT
         CLI   GETFCT,C'G'        WAS DATDSPLY CALLED BY GET?
         BNE   DSPL1              NO, GO FOR MESSAGE INTERPRETATION
         CLC   RECPRFXL(4,BUFPTR),=C'* 12'  IS THE CURRENT CARD NO.12
         BNE   DSPL1X             NO, GO TO DISPLAY DATA RECORD
         MVC   PWRRECNO,RECLOGNO  SAVE LOGICAL RECORD NUMBER
         B     DSPL1X             GO TO DISPLAY DATA RECORD
         SPACE 1
DSPL1    DS    0H
         CLI   RECTYPE,RECTFIXM   FIXED FORMAT MSG RECORD RETURNED ?
         BNE   DSPL1X             GO TO DISPLAY CONSOLE MESSAGE
         LH    R2,RECLNGTH        GET LENGTH OF FIRST/NEXT DATA REC.
         LA    BUFPTR,RECPRFXL(,BUFPTR)  POINT TO FIX FORMAT RECORD
         DROP  BUFPTR             RELEASE PREFIX ADDRESSABILITY
         USING PXFMDSCT,BUFPTR    MAKE FIX FORM MSG ADDRESSABLE
         MVC   FFMHD+15(L'JOBNAME),PXFMNAME   PASS NAME TO MSG SKELET.
         SR    R1,R1              CLEAR REGISTER
         ICM   R1,3,PXFMJNUM      PICK UP BINARY JOB NUMBER
         CVD   R1,HELP8           CONVERT TO PACKED DECIMAL
         UNPK  HELP5,HELP8+5(3)   UNPACK 3 DIGITS
         OI    HELP5+4,X'F0'      CHANGE X'C.' TO PRINTABLE X'F.'
         MVC   FFNHD+5(5),HELP5   PASS DEC. JOBNUMBER TO MSG SKEL
         MVC   FFCHD+7(1),PXFMCLSS     PASS JOB CLASS TO MSG SKELETON
         MVC   JOBQNUM,PXFMQNUM   SAVE INTERNAL Q-ENTRY-# FOR DIR. GET
         DROP  BUFPTR             RELEASE FF-MSG ADDRESSABILITY
         SPACE 1
         DPLAY FFDSPLY,72         DISPLAY ASSEMBLED DISPLAY LINE
         SPACE 1
         B     DSPL1Y             GO PROCESS NEXT/LAST PASSED RECORD
         SPACE 2
DSPL1X   DS    0H
         USING RECPRFIX,BUFPTR    MAKE RECORD PREFIX ADDRESSABLE
         LH    R2,RECLNGTH        GET LENGTH OF FIRST/NEXT DATA REC.
         LA    BUFPTR,RECPRFXL(,BUFPTR)     SKIP RECORD PREFIX
         DROP  BUFPTR             RELEASE PREFIX ADDRESSABILITY
         SPACE 1
         DPLAY (BUFPTR),(R2)      DISPLAY CURRENT DATA RECORD
         SPACE 2
```

```
DSPL1Y   DS    0H                                                       00879664
         LA    R1,RECPRFXL(,R2)    CALC. LENGTH OF RECORD INCL. PREFIX  00879672
         SR    BUFLN,R1            CALC.LENGTH OF DATA STILL IN BUFFER  00879680
         LA    BUFPTR,0(R2,BUFPTR)     POINT TO NEXT RECORD             00879688
         LTR   BUFLN,BUFLN         ALL DATA IN BUFFER DISPLAYED?        00879696
         BNZ   DSPL0               NO, GO TO DISPLAY NEXT DATA REC.     00879704
         SPACE 1                                                        00879712
         CLI   PXPFBKCD,PXP00EOD   END OF DATA?                         00879720
         BER   RE                  YES, RETURN TO CALLER                00879728
         SPACE 1                                                        00879736
*   IF THIS ROUTINE IS CALLED BY THE GET FUNCTION, 'SEND (MORE) DATA'   00879744
*   HAS TO BE INDICATED IN THE ACTION BYTE. IN ALL OTHER CASES          00879752
*   'RETURN (MORE) MESSAGES' MUST BE SET.                               00879760
         SPACE 1                                                        00879768
DSPL2    DS    0H                                                       00879776
         XC    IJBXBLN,IJBXBLN     INDICATE ZERO BUFFER LENGTH          00879784
         MVI   PXUBTYP,0            CLEAR BUFFER TYPE BYTE              00879792
         MVI   PXUACT1,PXUATRMR    INDICATE A 'RETURN MESSAGE' REQUEST  00879800
         CLI   GETFCT,C'G'          WAS DATDSPLY CALLED BY GET?         00879808
         BNE   DSPL3                NO, KEEP RETURN MESSAGE INDICATION  00879816
         MVI   PXUACT1,PXUATSDR    INDICATE A 'SEND DATA' REQUEST       00879824
DSPL3    DS    0H                                                       00879832
         MVC   FAILLABL,=C'DSPL2 ' INSERT CODE LABEL FOR DIAGNOSTIC     00879840
         BAL   RD,SENDR            GO TO SENDR ROUTINE                  00879848
         CLI   PXPRETCD,PXPRCOK    WAS VSE/POWER RETURNCODE ZERO?       00879856
         BNE   REQFAIL             NO, GO TO HANDLE REQUEST FAILURE     00879864
         B     DATDSPLY            YES, START DISPLAYING AGAIN          00879872
         EJECT                                                          00879880
*********************************************************************** 00879888
**      >> ROUTINE TO HANDLE REQUEST FAILURES         <<             ** 00879896
**  THE ROUTINE IS CALLED IF VSE/POWER RC/FBKC WAS NOT ZERO          ** 00879904
*********************************************************************** 00879912
         SPACE 1                                                        00879920
REQFAIL  DS    0H                                                       00879928
         MVC   FAILFUNC,=C'SENDR   ' INSERT FAILING FUNCTION INTO MSG   00879936
         BAL   RE,MSGRCFB          PREPARE RC/FBKC DISPLAY              00879944
         BAL   RE,MSGDSPLY         DISPLAY MESSAGE ON CONSOLE           00879952
         B     FINDUMP             GO TO TERMINATION WITH DUMP          00879960
         EJECT                                                          00879968
*********************************************************************** 00880000
**     >>       MESSAGE BUILD ROUTINE FOR FAILMSG            <<      ** 00881000
** BRANCHED TO FROM ANY CALLER TO FILL SELECTED FIELDS OF THE DIAG-  ** 00882000
** NOSTIC MESSAGE.  RETURNS TO CALLER VIA REGISTER 14 (RE).          ** 00883000
*********************************************************************** 00884000
         SPACE 1                                                        00885000
MSGRETC  DS    0H                                                       00886000
         UNPK  HELP,IJBXRETC(2)        UNPACK HEX XPCC RETURN CODE      00887000
         TR    HELP(2),TRTAB           CONVERT TO PRINTABLE HEX-VALUE   00888000
         MVC   FAILRETC,HELP           INSERT PRINTABLE XPCC RET. CODE  00889000
         BR    RE                      RETURN TO CALLER                 00890000
         SPACE 1                                                        00891000
MSGREAS  DS    0H                                                       00892000
         UNPK  HELP,IJBXREAS(2)        UNPACK HEX XPCC REASON CODE      00893000
         TR    HELP(2),TRTAB           CONVERT TO PRINTABLE HEX-VALUE   00894000
         MVC   FAILREAS,HELP           INSERT PRINTABLE XPCC REAS. CODE 00895000
         BR    RE                      RETURN TO CALLER                 00896000
         SPACE 1                                                        00897000
MSGRCFB  DS    0H                                                       00898000
         UNPK  HELP,PXPRETCD(2)        UNPACK HEX VSE/POWER RETURN CODE 00899000
         TR    HELP(2),TRTAB           CONVERT TO PRINTABLE HEX-VALUE   00900000
         MVC   FAILPWRC,HELP           INSERT PRINTABLE POWER RET.CODE  00901000
         SPACE 1                                                        00902000
         UNPK  HELP,PXPFBKCD(2)        UNPACK HEX POWER FEEDBACK CODE   00903000
         TR    HELP(2),TRTAB           CONVERT TO PRINTABLE HEX-VALUE   00904000
         MVC   FAILPWFB,HELP           INSERT VSE/POWER FEEDACK CODE    00905000
         SPACE 1                                                        00905310
         CLC   FAILLABL,=C'GETBB1'     CALLED FROM 'DIRECT' GET, WHERE  00905320
```

```
*                                      FEEDBACK CODE 2 IS MEANINGFUL ?  00905330
         BNE   MSGRCFB2                NO, CONTINUE                     00905340
         CLI   PXPRETCD,PXPRCOKF       VSE/POWER RETCD = X'04' ?        00905350
         BNE   MSGRCFB2                NO, CONTINUE                     00905360
         CLI   PXPFBKCD,PXP04NOF       VSE/POWER FEEDBACK-1 = X'01' ?   00905370
         BNE   MSGRCFB2                NO, CONTINUE                     00905380
         UNPK  HELP,PXPFBKC2(2)        UNPACK HEX POWER FEEDBACK-2 CODE 00905390
         TR    HELP(2),TRTAB           CONVERT TO PRINTABLE HEX-VALUE   00905400
         MVC   FAILPWF2,HELP           INSERT VSE/POWER FEEDACK-2 CODE  00905410
MSGRCFB2 DS    0H                                                       00905420
         SPACE 1                                                        00905430
*   FOR RETC/FBKCD CODE PXP08CON (08/22) OR PXP08ROS (08/25) GENERALLY, 00905440
*   CONSIDER TO DISPLAY ALSO THE FEEDBACK-2 CODE FROM PXPFBKC2.         00905600
         BR    RE                      RETURN TO CALLER                 00906000
         SPACE 1                                                        00907000
MSGDSPLY DS    0H                                                       00908000
         DPLAY FAILMSG,72              DISPLAY FAILURE MESSAGE          00909290
         MVC   FAILMSG,FAILCOPY        REFRESH FAILURE MESSAGE          00909580
         SPACE 1                                                        00910000
         BR    RE                                                       00911000
         EJECT                                                          00912000
*********************************************************************** 00913000
**           >> CENTRAL XPCC SENDR ROUTINE  <<                       ** 00914000
**  BEFORE THIS ROUTINE IS CALLED, THE PROGRAM INSERTS THE CALLING   ** 00915000
**  POINT IN THE DIAGNOSTIC MESSAGE THAT IS ISSUED SHOULD THE SENDR  ** 00916000
**  MACRO FAIL.  THIS ROUTINE:                                       ** 00917000
**   - ISSUES THE XPCC MACRO WITH FUNC=SENDR AND WAITS FOR THE       ** 00918000
**     SECB TO BE POSTED.  IT CHECKS REGISTER 15 (RF) AND THE VSE    ** 00919000
**   - CHECKS REGISTER 15 (RF) AND THE z/VSE RETURN- AND REASON CODES  ** 00920000
**     IN FIELDS IJBXRETC AND IJBXREAS, RESPECTIVELY.                ** 00921000
**   - CHECKS THE VSE/POWER RETURN CODE IN FIELD PXPRETCD IF         ** 00922000
**     VSE/POWER DISCONNECTS THE COMMUNICATION PATH WITH A PURGE.    ** 00923000
**  THE ROUTINE RETURNS TO THE CALLER IF THE XPCC MACRO CALL COM-    ** 00924000
**  PLETED SUCCESSFULLY OR, IN CASE OF A FAILURE, THE VSE/POWER RE-  ** 00925000
**  TURN CODE IS NOT TOO SEVERE.  RETURN IS PROVIDED VIA REGISTER    ** 00926000
**  13 (RD).                                                         ** 00927000
*********************************************************************** 00928000
         SPACE 1                                                        00929890
*        REGISTER USAGE FOR SENDR ROUTINE                               00930780
         SPACE 2                                                        00931670
*        R3 - WORK REGISTER (FOR WAIT)                                  00932560
*        RD - REGISTER USED TO RETURN TO CALLER                         00933450
*                                                                       00934340
*        CALLED FROM: PUT REQUEST TO RDR QUEUE                          00935230
*                     CTL REQUEST                                       00936120
*                     GET REQUEST                                       00937010
*                     PUT REQUEST TO LST QUEUE                          00937900
*                     DATDSPLY ROUTINE                                  00938790
*                                                                       00939680
*        EXIT TO CALLER (SEE COMMENT ABOVE)                             00940570
*                OR TO DISCT OR FINDUMP IN CASE OF A FAILURE            00941460
*                                                                       00942350
         SPACE 2                                                        00943240
         SPACE 1                                                        00945000
SENDR    DS    0H                                                       00946000
         XPCC  XPCCB=(R4),FUNC=SENDR      SEND BUFFER TO VSE/POWER      00947000
         LTR   RF,RF               DID WE GET A ZERO RETURN CODE ?      00948000
         BZ    WAITSECB            ..YES, THEN WAIT FOR REPLY OF POWER  00949000
         SPACE 2                                                        00950000
*   IF THE SENDR MACRO COMPLETES WITH RF=X'08', THEN THE ROUTINE:       00951000
*   1.  FILLS THE DIAGNOSTIC MESSAGE ACCORDING TO THE VSE RETURN CODE.  00952000
*   2.  DISPLAYS THE MESSAGE.                                           00953000
*   3.  TERMINATES WITH OR WITHOUT A DUMP.                              00954000
*   THERE IS NO RETURN TO THE CALLER OF SENDR.                          00955000
         SPACE 1                                                        00956000
TESTRETC DS    0H                                                       00957000
         CLI   IJBXRETC,IJBXNOC3   DID VSE/POWER ABNORMALLY TERMINATE ? 00958000
```

```
         BE    ABNPOW             ..YES, THEN GO TO STOP PWRSASEX
         MVC   FAILFUNC,=C'SENDR  '    INSERT 'SENDR ' INTO MSG TEXT
         BAL   RE,MSGRETC         PUT XPCC RETURN CODE INTO MSG
         CLI   IJBXRETC,IJBXNOC2  DID VSE/POWER GIVE A DISCONNECT PURGE
         BE    TERMCONN           ..YES,THEN GO TO SHOW WHY, TERMINATE
         BAL   RE,MSGDSPLY        DISPLAY DIAGNOSTIC MESSAGE ON CONS.
         B     FINDUMP            TERMINATE PWRSASEX WITH PART.DUMP
         SPACE 1
ABNPOW   DS    0H
         DPLAY FAILM2,72
         SPACE 1
         B     DISCT              DISCONN AND TERMIN XPCC LINK, EOJ
         SPACE 2
*   THE ROUTINE WAITS FOR THE SEND ECB TO BE POSTED.  IT RETURNS TO THE
*   CALLER IF THE SYSTEM PASSED A REASON CODE OF ZERO, THAT IS, THE
*   XPCC CONNECTION IS ERROR FREE.
*   FOR A NON-ZERO REASON CODE, THE ROUTINE DISPLAYS A DIAGNOSTIC
*   MESSAGE AND TERMINATES WITH OR WITHOUT A DUMP.
         SPACE 1
WAITSECB DS    0H
         LA    R3,IJBXSECB        LOAD ADDRESS OF SEND COMPLETION ECB
         WAIT  (R3)               WAIT FOR COMPLETION  OF SENDR
         CLI   IJBXREAS,REASOK    DID ANY CONNECTION ERROR OCCUR ?
         BER   RD                 .. NO, THEN RETURN TO CALLER
         SPACE 1
BADREAS  DS    0H
         TM    IJBXREAS,IJBXABDC  DID VSE/POWER TERMINATE ABNORMALLY ?
         BO    ABNPOW             .. YES, GIVE MESSAGE AND GO TO EOJ
         MVC   FAILFUNC,=C'SENDR  '    INSERT 'SENDR ' INTO MSG TEXT
         BAL   RE,MSGREAS         FILL XPCC REASON CODE INTO MSG
TERMCONN DS    0H
         BAL   RE,MSGRCFB         PUT VSE/POWER RETURN/FEEDBACK TO MSG
         BAL   RE,MSGDSPLY        DISPLAY DIAGNOSTIC MESSAGE
         CLI   PXPRETCD,PXPRCPVL  VSE/POWER RC = PROTOCOL VIOLATION?
         BE    FINDUMP            .. YES, USER ERROR
         B     DISCT              SYSTEM ERROR OCCURED        @D23QDIR
         EJECT
***********************************************************************
**                 D E F I N I T I O N S                           **
***********************************************************************
         SPACE 2
TRTAB    EQU   *-240              ENTRY POINT FOR TRANSLATE TABLE
         DC    X'F0F1F2F3F4F5F6F7F8F9C1C2C3C4C5C6' TRANSLATE TABLE
         SPACE 1
EIGHTDC  DC    X'08'              BYTE TO TEST RETURN CODE
HELP     DC    CL3' '             FIELD FOR UNPACK RET CODE
HELP5    DS    CL5                FIELD FOR UNPACK JOBNUMBER
         DS    0D                 ENFORCE ALIGNMENT FOR 'CVD' FIELD
HELP8    DS    CL8                FIELD FOR CONVERT DECIMAL JOBNUMBER
         SPACE 1
WAITLIST DC    A(INTECB)          INTECB = 1ST ELEMENT OF WAITLIST
LISTCECB DC    A(0)               IJBXCECB = 2ND ELEM. OF WAITLIST
LISTEND  DC    X'FF'              INDICATE END OF WAITLIST
         SPACE 1
INTECB   DS    F                  ECB USED TO WAIT FOR TIMER INTERVALS
         SPACE 1
EIGHT    EQU   X'08'              RETURN CODE X'08'
POSTBIT  EQU   X'80'              MASK FOR A POSTED ECB
REASOK   EQU   X'00'              ZERO VSE/AF REASON CODE
         EJECT
***********************************************************************
*          DEFINITIONS FOR PUT,CTL AND GET REQUEST                  *
***********************************************************************
         SPACE 1
M1       EQU   1                  MASK BIT SETTING
M7       EQU   7                  MASK BIT SETTING
ZERO     EQU   0
```

```
ONE      EQU   1                                                01020720
         SPACE 1                                                01021310
NOOFRECS EQU   7                  NUMBER OF RECORDS IN A SEGMENT      01021900
DISP     EQU   C'L'               DISPOSITION OF OUTPUT TO BE SENT    01022490
PRIOR    EQU   C'9'               PRIORITY OF OUTPUT TO BE SENT       01023080
SEPPAGE  EQU   4                  NUMBER OR SEPARATOR PAGES/CARDS     01023670
         SPACE 1                                                01024260
RECORDCT EQU   2                  USE R2 AS REC COUNTER IN LOOP       01024850
BUFPTR   EQU   10                 USE RA AS BUFPOINTER                01025440
DATAPTR  EQU   11                 USE RB AS DATA POINTER              01026030
BUFLN    EQU   12                 USE RC TO CALC REMAINING BUFLEN     01026620
         SPACE 2                                                01027210
JOBNAME  DS    CL8                FIELD TO SAVE JOBNAME RET'D BY POW. 01027800
JOBNUM   DS    XL2                FIELD TO SAVE JOBNUMB.RET'D BY POW. 01028390
PWRRECNO DS    F                  FIELD TO SAVE POW. LOGICAL REC NO.  01028980
JOBQNUM  DS    F                  FIELD TO SAVE INT. QUEUE ENTRY NO.  01029570
ZERONUM  DC    H'0'               TO INDICATE 'NO JOBNUMBER SPECIFIED' 01030160
*                                 BECAUSE SPLXQNUM IS 'UNIQUE'        01030750
GETFCT   DC    C' '               FIELD TO IDENTIFY GET AS CALLER OF  01031340
*                                                         DATDSPLY    01031930
         SPACE 1                                                01032520
FORMS    DC    CL8'AABB'          FORMS OF OUTPUT TO BE SENT          01033110
JOBNLAB  DC    CL8'EXAMPSEG'      NAME OF OUTPUT TO BE SPOOLED        01033700
         SPACE 1                                                01034290
JOBCMD   DS    0CL130             COMMAND AT ITS MAXIMUM LENGTH, TER- 01034880
*                                 MINATED WITH AT LEAST ONE BLANK(!)  01035470
CMDHEAD  DC    CL13'PDISPLAY LST,'  FIXED START OF COMMAND            01036060
CMDBODY  DC    CL117' '             DYNAMIC BODY/END OF COMMAND       01036650
         SPACE 1                                                01037240
CMDCLAS  DC    C'CCLASS=S '       SELECTION CLASS FOR PDISPLAY        01037830
         SPACE 1                                                01038420
FFDSPLY  DS    0CL72              MESSAGE EXTRACT FROM FF MSG RECORD  01039010
FFMHD    DC    CL23'1QSAS  LST JNM=        '   MSG HEADER PLUS JOBNAME 01039600
FFNHD    DC    CL10' JNB=     '                JOBNUMBER ONLY         01040190
FFCHD    DC    CL8' CLASS= '                   JOBCLASS ONLY          01040780
         DC    CL31' '                         BLANK MESSAGE TRAILER  01041370
         EJECT                                                  01041960
*********************************************************************** 01044000
*           MESSAGE AREA FOR FAILING MACRO CALLS                      * 01045000
*********************************************************************** 01046000
         SPACE 1                                                01047000
FAILMSG  DS    0CL72                                            01048000
F1       DC    C'FUNC='                                         01049000
FAILFUNC DC    CL8' '             REQUESTED FUNCTION                  01050000
F2       DC    C' FAILED AT: '                                  01051000
FAILLABL DC    CL6' '             CODE LABEL OF FAILING FUNCTION      01052590
F3       DC    C' XPCC='                                        01053180
FAILRETC DC    CL2'00'            RETURN CODE RECEIVED IN IJBXRETC    01054000
F4       DC    C'/'                                             01055000
FAILREAS DC    CL2'00'            REASON CODE RECEIVED IN IJBXREAS    01056000
F5       DC    C' PWR-RC/FB1/FB2='                              01057490
FAILPWRC DC    CL2'00'            VSE/POWER RETURN CODE IN IJBXRUSR   01058000
F6       DC    C'/'                                             01059000
FAILPWFB DC    CL2'00'            POWER FEEDBACK CODE 1 IN IJBXRUSR   01060190
F7       DC    C'/'                                             01060380
FAILPWF2 DC    CL2'00'            VSE/POWER FEEDBACK CODE 2 IN IJBXRUSR01060570
F8       DC    CL6' '                                           01060760
         SPACE 1                                                01060950
FAILCOPY DS    CL72                                             01061140
         SPACE 1                                                01062000
FAILM1   DC    CL72'VSE/POWER ALREADY IN TERMINATION, NO MORE CONNECTIO*01063000
               N REQUEST ACCEPTED'                              01064000
FAILM2   DC    CL72'VSE/POWER ABNORMAL TERMINATION, CONNECTION DISRUPTE*01065000
               D'                                               01066000
FAILM3   DC    CL72'CONNECTION COULD NOT BE COMPLETED WITHIN 2 MINUTES' 01067000
FAILM4   DC    CL72'LIST QUEUE ENTRY COULD NOT BE FOUND, PWRSASEX WILL *01068000
               STOP'                                            01069000
```

```
         SPACE 1                                                        01070000
SUCCM1   DC    CL72'>>> XPCC CONNECTION TO VSE/POWER SUCCESSFULLY BUILT*01071000
                <<<'                                                    01072000
SUCCM1A  DC    CL72'>>> PWRSASEX WILL SUBMIT JOB ''EXAMPLE'' FOR EXECUT*01073390
               ION IN CLASS 4 (!) <<<'                                  01073780
SUCCM1B  DC    CL72'>>> PWRSASEX WILL WAIT FOR EXECUTION OF JOB ''EXAMP*01074170
               LE'' IN CLASS 4 <<<'                                     01074560
SUCCM1C  DC    CL72'>>> NOW ISSUE PDISPLAY FOR OUTPUT ''EXAMPLE'' AND P*01074950
               ASS IT TO CONSOLE <<<'                                   01075340
SUCCM2   DC    CL72'>>> NOW PWRSASEX WILL RESTART ON RECORD NO.12 <<<'  01075730
SUCCM3   DC    CL72'>>> THE VSE/POWER LIST QUEUE MUST NOW CONTAIN 3 RBS*01076120
               -LIKE SEGMENTS ... <<<'                                  01076510
SUCCM4   DC    CL72'>>> ... NAMED ''EXAMPSEG'' AND A SINGLE ENTRY NAMED*01076900
                ''EXAMPLE'' <<<'                                        01077290
SUCCM6   DC    CL72'>>> *** SUCCESSFUL TERMINATION OF PWRSASEX *** <<<' 01078000
SUCCM7   DC    CL72'>>> NOW FOLLOWS THE DISPLAY OF THE LIST ENTRY ''EXA*01079590
               MPLE'' <<<'                                              01080180
SUCCM8   DC    CL72'>>> NEXT PWRSASEX WILL SUBMIT DATA TO THE LIST QUEU*01081000
               E <<<'                                                   01082000
         EJECT                                                          01083000
*********************************************************************** 01084000
*         JOB 'EXAMPLE' TO BE PASSED TO VSE/POWER                     * 01085000
*********************************************************************** 01086000
         SPACE 1                                                        01087000
JECL1    DC    C'* $$ JOB JNM=EXAMPLE,DISP=K,CLASS=4                   '  01088690
JECL2    DC    C'* $$ LST LST=SYSLST,DISP=K,CLASS=S                    '  01089490
JCL1     DC    C'// JOB EXAMPLE                                        '  01090000
DATACARD DC    C'* 01---------------------*----------------------01 *'  01091000
         DC    C'* 02-------------------*    *-------------------02 *'  01092000
         DC    C'* 03-----------------*        *-----------------03 *'  01093000
         DC    C'* 04---------------*            *---------------04 *'  01094000
         DC    C'* 05-------------*                *-------------05 *'  01095000
         DC    C'* 06-----------*                    *-----------06 *'  01096000
         DC    C'* 07---------*                        *---------07 *'  01097000
         DC    C'* 08-------*                            *-------08 *'  01098000
         DC    C'* 09-----*                                *-----09 *'  01099000
         DC    C'* 10---*                                    *---10 *'  01100000
         DC    C'* 11-*                                        *-11 *'  01101000
         DC    C'* 12---*                                    *---12 *'  01102000
         DC    C'* 13-----*                                *-----13 *'  01103000
         DC    C'* 14-------*                            *-------14 *'  01104000
         DC    C'* 15---------*                        *---------15 *'  01105000
         DC    C'* 16-----------*                    *-----------16 *'  01106000
         DC    C'* 17-------------*                *-------------17 *'  01107000
         DC    C'* 18---------------*            *---------------18 *'  01108000
         DC    C'* 19-----------------*        *-----------------19 *'  01109000
         DC    C'* 20-------------------*    *-------------------20 *'  01110000
         DC    C'* 21---------------------*----------------------21 *'  01111000
JCL2     DC    C'/&&                                                    ' 01112000
JECL3    DC    C'* $$ EOJ                                              '  01113000
ENDIND   DC    C'/+'                                                    01114000
         EJECT                                                          01115000
*********************************************************************** 01116000
*        CROSS PARTITION CONTROL BLOCK                                * 01117000
*********************************************************************** 01118000
         SPACE 1                                                        01119000
OWNXPCCB XPCCB APPL=PWRSASEX,TOAPPL=SYSPWR,                            *01120000
               BUFFER=(SENDBUF,400),REPAREA=(REPLBUF,500)               01121000
         SPACE 2                                                        01122000
*********************************************************************** 01123000
*        STORAGE RESERVATION FOR  XPCC SEND AND REPLY BUFFER          * 01124000
*********************************************************************** 01125000
         SPACE 1                                                        01126000
SENDBUF  DS    CL400              BUFFER USED FOR XPCC SENDR TO POWER   01127000
LASTPREC DC    A(SENDBUF+L'SENDBUF-RECPRFXL-L'DATACARD) LAST POSSIBLE   01128000
*                                 RECORD THAT FITS INTO SEND BUFFER     01129000
REPLBUF  DS    CL500              BUFFER FOR RECEIPT OF DATA FROM POWER 01130000
```

```
         EJECT
***********************************************************************
         LTORG
***********************************************************************
         EJECT
***********************************************************************@
**      >>         GENERATE   S P L                <<              *@
**       THIS SPL IS LATER ON UPDATED IN ORDER TO INDICATE A        *@
**       PUT, CTL, OR GET REQUEST WITH THE DESIRED PARAMETERS       *@
***********************************************************************@
         SPACE 1                                                     @
OWNSPL   PWRSPL TYPE=GEN,USERID=SASUSER1,PRFX=OWN                    @
         EJECT                                                       @
***********************************************************************
*        DUMMY SECTION OF  VSE/POWER SPOOL PARAMETER LIST (SPL)      *
***********************************************************************
         SPACE 1
OWNSPLDS PWRSPL TYPE=MAP
         EJECT
***********************************************************************
*        DUMMY SECTION OF  CROSS PARTITION CONTROL BLOCK  (XPCCB)    *
***********************************************************************
         SPACE 1
         MAPXPCCB
         EJECT
***********************************************************************
*        GENERAL EQUATES                                             *
***********************************************************************
         SPACE 1
R0       EQU   0                  WORK REGISTER
R1       EQU   1                  WORK REGISTER + USED BY PWRSPL MACRO
R2       EQU   2                  WORK REGISTER
R3       EQU   3                  WORK REGISTER
R4       EQU   4                  ADDR REG FOR XPCCB DSECT
R5       EQU   5                  ADDR REG FOR USER DATA TO BE SENT
R6       EQU   6                  ADDR REG FOR RECEIVED USER DATA
R7       EQU   7                  ADDR REG FOR SPL DSECT
R8       EQU   8                  FIRST BASE REGISTER OF PWRSASEX
R9       EQU   9                  SECOND BASE REGISTER OF PWRSASEX
RA       EQU   10                 WORK REGISTER
RB       EQU   11                 WORK REGISTER
RC       EQU   12                 WORK REGISTER
RD       EQU   13                 BRANCH AND LINK REGISTER FOR SENDR
RE       EQU   14                 BRANCH AND LINK REG. FOR DATDSPLY
RF       EQU   15                 MACRO CALL RETURN CODE REGISTER
         SPACE 1
```

Control Statements for Execution

```
* $$ JOB JNM=PWRSARUN,DISP=K,CLASS=A
// JOB PWRSARUN
// LIBDEF *,SEARCH=...
*
* PROVIDE ... LIB.SUBLIB USED IN JOB PWRSACAT
*
// EXEC PWRSASEX
/&
* $$ EOJ
```

For execution of phase PWRSASEX, submit this job planned for processing in class A. Remember to have a partition available (F4) to process the job in class 4. For details or changes, such as of class 4, refer to the header note in the PWRSASEX source code.

PRINTLOG of PWRSASEX Execution

```
R RDR,PWRSARUN
AR 0015 1C39I COMMAND PASSED TO VSE/POWER
F1 0001 1R88I  OK
BG 0001 1Q47I   BG PWRSARUN 00178 FROM PNET631 , TIME=17:28:23
BG 0000 // JOB PWRSARUN
        DATE 06/20/2000, CLOCK 17/28/23
BG 0000 >>> XPCC CONNECTION TO VSE/POWER SUCCESSFULLY BUILT <<<
BG 0000 >>> PWRSASEX WILL SUBMIT JOB 'EXAMPLE' FOR EXECUTION IN CLASS 4 (!) <<<
BG 0000 >>> PWRSASEX WILL WAIT FOR EXECUTION OF JOB 'EXAMPLE' IN CLASS 4 <<<
F4 0001 1Q47I   F4 EXAMPLE 00179 FROM PNET631(SASUSER1) , TIME=17:28:23
F4 0004 // JOB EXAMPLE
        DATE 06/20/2000, CLOCK 17/28/23
F4 0004 * 01---------------------*----------------------01 *
F4 0004 * 02-------------------*    *-------------------02 *
F4 0004 * 03-----------------*        *-----------------03 *
F4 0004 * 04---------------*            *---------------04 *
F4 0004 * 05-------------*                *-------------05 *
F4 0004 * 06-----------*                    *-----------06 *
F4 0004 * 07---------*                        *---------07 *
F4 0004 * 08-------*                            *-------08 *
F4 0004 * 09-----*                                *-----09 *
F4 0004 * 10---*                                    *---10 *
F4 0004 * 11-*                                        *-11 *
F4 0004 * 12---*                                    *---12 *
F4 0004 * 13-----*                                *-----13 *
F4 0004 * 14-------*                            *-------14 *
F4 0004 * 15---------*                        *---------15 *
F4 0004 * 16-----------*                    *-----------16 *
F4 0004 * 17-------------*                *-------------17 *
F4 0004 * 18---------------*            *---------------18 *
F4 0004 * 19-----------------*        *-----------------19 *
F4 0004 * 20-------------------*    *-------------------20 *
F4 0004 * 21---------------------*----------------------21 *
F4 0004 EOJ EXAMPLE
        DATE 06/20/2000, CLOCK 17/28/24, DURATION   00/00/00
F4 0001 1Q34I   F4 WAITING FOR WORK
BG 0000 >>> NOW ISSUE PDISPLAY FOR OUTPUT 'EXAMPLE' AND PASS IT TO CONSOLE <<<
BG 0000 1R46I    LIST QUEUE   P D C S  PAGES  CC FORM          -- CTL standard flow
BG 0000 1R46I  EXAMPLE  00179 3 K S    1   1    TO=(SASUSER1)  -- CTL standard flow
FROM=(SASUSER1)                                                -- CTL standard flow

BG 0000 1QSAS  LST JNM=EXAMPLE  JNB=00179 CLASS=S              -- CTL alternat.flow

BG 0000 >>> NOW FOLLOWS THE DISPLAY OF THE LIST ENTRY 'EXAMPLE' <<<
BG 0000 // JOB EXAMPLE
DATE 06/20/2000, CLOCK 17/28/23
BG 0000 * 01---------------------*----------------------01 *
BG 0000 * 02-------------------*    *-------------------02 *
BG 0000 * 03-----------------*        *-----------------03 *
BG 0000 * 04---------------*            *---------------04 *
BG 0000 * 05-------------*                *-------------05 *
BG 0000 * 06-----------*                    *-----------06 *
BG 0000 * 07---------*                        *---------07 *
BG 0000 * 08-------*                            *-------08 *
BG 0000 * 09-----*                                *-----09 *
BG 0000 * 10---*                                    *---10 *
BG 0000 * 11-*                                        *-11 *
BG 0000 * 12---*                                    *---12 *
BG 0000 * 13-----*                                *-----13 *
BG 0000 * 14-------*                            *-------14 *
BG 0000 * 15---------*                        *---------15 *
BG 0000 * 16-----------*                    *-----------16 *
BG 0000 * 17-------------*                *-------------17 *
BG 0000 * 18---------------*            *---------------18 *
BG 0000 * 19-----------------*        *-----------------19 *
```

```
BG 0000 * 20-------------------*    *-------------------20 *
BG 0000 * 21--------------------*---------------------21 *
BG 0000 EOJ EXAMPLE
DATE 06/20/2000, CLOCK 17/28/24, DURATION   00/00/00
BG 0000 >>> NOW PWRSASEX WILL RESTART ON RECORD NO.12 <<<
BG 0000 * 12---*                                   *---12 *
BG 0000 * 13-----*                               *-----13 *
BG 0000 * 14-------*                           *-------14 *
BG 0000 * 15---------*                       *---------15 *
BG 0000 * 16-----------*                   *-----------16 *
BG 0000 * 17-------------*               *-------------17 *
BG 0000 * 18---------------*           *---------------18 *
BG 0000 * 19-----------------*       *-----------------19 *
BG 0000 * 20-------------------*    *-------------------20 *
BG 0000 * 21--------------------*---------------------21 *
BG 0000 EOJ EXAMPLE
DATE 06/20/2000, CLOCK 17/28/24, DURATION   00/00/00
BG 0000 >>> NEXT PWRSASEX WILL SUBMIT DATA TO THE LIST QUEUE <<<
BG 0000 >>> THE VSE/POWER LIST QUEUE MUST NOW CONTAIN 3 RBS-LIKE SEGMENTS ... <<
BG 0000 >>> ... NAMED 'EXAMPSEG' AND A SINGLE ENTRY NAMED 'EXAMPLE' <<<
BG 0000 >>> *** SUCCESSFUL TERMINATION OF PWRSASEX *** <<<
BG 0000 EOJ PWRSARUN
        DATE 06/20/2000, CLOCK 17/28/38, DURATION   00/00/15
BG 0001 1Q34I   BG WAITING FOR WORK
D LST,*EXAMP
AR 0015 1C39I COMMAND PASSED TO VSE/POWER
F1 0001 1R46I    LIST QUEUE   P D C S  PAGES  CC FORM
F1 0001 1R46I  EXAMPLE  00179 3 K S       1   1      TO=(SASUSER1)
                                                             FROM=(SASUSER1)
F1 0001 1R46I  EXAMPSEG 00182 9 L Z       1   1 AABB S=001 FROM=(SASUSER1)
F1 0001 1R46I  EXAMPSEG 00182 9 L Z       1   1 AABB S=002 FROM=(SASUSER1)
F1 0001 1R46I  EXAMPSEG 00182 9 L Z       1   1 AABB S=003 FROM=(SASUSER1)
```

This printlog can help you synchronize the intended source code actions of PWRSASEX with the actual execution steps logged on the console. PWRSASEX executes in BG (class A) and

- Writes ">>>" progress messages to the console from BG
- Submits job EXAMPLE, which executes in F4, producing Job Control comment lines on the console.
- Produces a list queue display on the console
- Retrieves data of this list queue entry for display by the BG partition.
- Submits list queue output segments, which are finally verified by a PDISPLAY LST,*EXAMP command entered by the operator.

Chapter 14. Return and Feedback Codes and Their Meanings

The following figure lists

- the mnemonics of the feedback code (containing the respective return code in position 4 + 5)
- the hexadecimal values of the return code
- the hexadecimal values of the feedback code
- and the meaning of the feedback code.

The following abbreviation is used for groups of allowed characters:

```
alpham   =  A-Z 0-9 $ @ #
alpham*  =  A-Z 0-9 $ @ # *
alphaj   =  A-Z 0-9 $ @ # . / -
alphaj*  =  A-Z 0-9 $ @ # . / - *
alphajb  =  A-Z 0-9 $ @ # . / - blank
alphap   =  A-Z $ @ #
```

Table 80. Return and Feedback Codes and Their Meanings

Mnemonic PXPFBKCD	Return Code	Fdbk Code	Meaning
PXP00OK	00	00	OK
PXP00EOD	00	01	End of Data, end of messages, or no messages at all - The buffer passed to the application program contains the last data or message record.
PXP00NJB	00	02	EOD received but not on job boundary - End of Data was indicated in the action byte PXUACT1 for a PUT Job Service, but the job does not end with a /& or an * $$ EOJ statement.
PXP00NRS	00	03	No records spooled - One of the following actions is indicated in action byte PXUACT1 for a PUT Job/Output Service, but no records are spooled since last Open request: - End of Data - Quit request - Checkpoint request - Segment request

Table 80. Return and Feedback Codes and Their Meanings (continued)

Mnemonic PXPFBKCD	**Return Code**	**Fdbk Code**	**Meaning**
PXP00RTR	00	04	Record truncated - The record length of a record passed to VSE/POWER is greater than the maximum allowed record length specified in SPLDLREC. VSE/POWER passes the offset from the beginning of user program's send buffer to the truncated record back in field PXPROFF of the user data. If more than one record was truncated, the offset is always for the last truncated record.
PXP00ZBF	00	05	Zero data buffer received - The buffer received by VSE/POWER does not contain any data record. It is empty.
PXP00CIA	00	06	Checkpoint record number changed - PUT Output Service: A Restart request is indicated in the user data for an already checkpointed output entry. Restart means that spooling continues at the specified record number. If this record number is less than the checkpoint record number, VSE/POWER uses the restart record number as new checkpoint number.
PXP00NCM	00	07	PUT-OPEN service: Job completion message retrieval support not available. The message queue size has been defined with SET JCMQ=0 during VSE/POWER startup.
PXP00LCM	00	08	PUT-OPEN service: The capacity of the message queue identified by your userid and applid is nearly exhausted. Space for 2 - 5 messages is left in the queue.
PXP00OCM	00	09	PUT-OPEN service: The capacity of the message queue identified with your tag is reached. Space for at most 1 message is left in the queue.

Table 80. Return and Feedback Codes and Their Meanings (continued)

Mnemonic PXPFBKCD	Return Code	Fdbk Code	Meaning
PXP04NOF	04	01	Job/Output queue entry not found - The job or output queue entry specified in the GET/CTL Open SPL was not found in the queue, or you are not entitled to access the queue entry because of non-matching user ID or even node ID. Refer also to "Scope of GET/CTL Access to Queue Entries" on page 61. If you issued a 'Direct' CTL request, refer to additional feedback-2 codes in Table 23 on page 73. If you issued a 'Direct' GET request, refer to additional feedback-2 codes in Table 30 on page 100. Or you issued a 'delete checkpoint information' request and the queue entry number which identifies the job does not match the characteristics specified in the SPL. See also Table 21 on page 70.
PXP04JOP	04	02	Job/Output queue entry protected - The password specified in the Open SPL does not match the password of the job/output for which the Open was done.
PXP04BSY	04	03	Job/Output queue entry marked active - The job or output queue entry specified in the SPL for a - GET Open for update request or + PUT Open restart request or + 'Direct' CTL Open (PALTER/PDELETE/PHOLD/PRELEASE) request + CTL Open for Delete Checkpoint request is currently processed by VSE/POWER. - GET Open for browse is currently processed by non-browse mode requests, or the maximum number of 255 (non-shared) or 15 (shared, per Sysid)parallel 'browses' has already been reached. Therefore, the entry is not accessible.
PXP04NDS	04	04	Job/Output queue entry not in dispatchable state - The job or output queue entry specified in the SPL for a GET Open request does not have disposition K or D or the job specified in the SPL for a GET Open request has time event scheduling operands. Consider using GET-Open for browsing instead.
PXP04IDP	04	05	Output queue entry not in appendable state - The output queue entry specified in the SPL for a PUT Open Append request has not disposition A.

Table 80. Return and Feedback Codes and Their Meanings (continued)

Mnemonic PXPFBKCD	Return Code	Fdbk Code	Meaning
PXP04RER	04	06	Restart error, record, line or page number out of range - The specified restart record/line/page number is greater than the number of records/lines/pages spooled by VSE/POWER.
PXP04CER	04	07	Checkpoint error, checkpoint number out of range - The specified checkpoint number is greater than the number of the record VSE/POWER has passed as last record during processing of a GET Data request.
PXP04SOD	04	08	Short on spool file space - VSE/POWER wants to write the data records received with PUT Data requests to the spool file but the spool file is full.
PXP04SOA	04	09	Short on account file space - VSE/POWER wants to write a spool account record for a spool (PUT Service) or retrieve (GET Service) operation when it is finished, but the account file is full.
PXP04BER	04	0A	Request prohibited in BROWSE mode - The following requests are not allowed for a queue entry which was accessed by a GET Open BROWSE request: - Close request - Purge request - Flush Hold request - Printing/Punch Failed request - Checkpoint request - Modify OPTB request if the caller is not CICS.
PXP04DNF	04	0B	Nothing found in specified queue(s) - A PDISPLAY queue command was passed with CTL Service toVSE/POWER , but no queue entry was eligible for display. If you issued a 'Direct' PDISPLAY CTL request, refer to additional feedback-2 codes Table 23 on page 73.
PXP04TQN	04	0C	Temporary queue entry not found - For a PDISPLAY queue command passed with the CTL Service to VSE/POWER, VSE/POWER builds a temporary LST queue entry containing the result of the display command. If this queue entry is not found 04/0C is passed back. This should never occur.

Table 80. Return and Feedback Codes and Their Meanings (continued)

Mnemonic PXPFBKCD	Return Code	Fdbk Code	Meaning
PXP04NMU	04	0D	No matching user id specified - The userid specified in the SPL for a PUT Open Restart or PUT Open Append request does not match the user id of the specified queue entry, or the output was created on another node.
PXP04WDP	04	0E	Invalid disposition - PUT Open Restart is not allowed for a queue entry with a disposition of A.
PXP04JSR	04	0F	Job suffix number mandatory - A PUT Open Restart request without a suffix number specified was passed to VSE/POWER, but the output queue entry consists of multiple segments.
PXP04NOQ	04	10	No order/signal queued - VSE/POWER received a 'return order' request from a DDS but there is currently no order or signal queued.
PXP04ONF	04	11	OPTB(s) not found - VSE/POWER received a Get-OPTB request during PUT or GET Service, but the specified OPTB(s) is not found or the queue entry does not have OTPBs.
PXP04NJC	04	12	GCM-OPEN service: No job completion message retrieval has been generated. The message queue size has been specified with SET JCMQ=0 in the VSE/POWER autostart procedure.
PXP04CKN	04	13	No extended checkpoint information exists. A request 'retrieve extended checkpoint information' is issued, but no checkpoint with extended information has been recorded (or already been deleted).
PXP04CKE	04	14	Extended checkpoint information exists, but can not be read. A request 'retrieve extended checkpoint information' is issued, but no extended information is available due to an I/O error.
PXP04NCK	04	15	No checkpoint information exists. A request 'delete checkpoint information' is issued, but no checkpoint has been recorded (neither with or without extended checkpoint information) or already been deleted.

Table 80. Return and Feedback Codes and Their Meanings (continued)

Mnemonic PXPFBKCD	Return Code	Fdbk Code	Meaning
PXP04NMF	04	16	GCM-OPEN service: No job completion message was found. You addressed a message queue by your applid and userid in which VSE/POWER could not find the messages you want to retrieve.
PXP04SAC	04	17	Job/output Spool Access Protection violation. VSE/POWER has been started with Spool Access Protection active. An XPCC SAS GET or SAS PUT-OUTPUT-APPEND/RESTART program attempted to access a spool entry, but the program's security logon user ID (either from the IBM Component terminal logon or the partition // ID or * $$ JOB SEC= statement) does not match the spool entry's authorized access user ID(s) (either the spool entry's origin or target user ID). The authorized access user IDs can be displayed with the PDISPLAY command (displayed as FROM= or TO=).
PXP04NAT	04	18	GET Restart to Active Record: Queue entry not active or is active on another shared system.
PXP04ANS	04	19	GET Restart to Active Record: Queue entry active by task which is not suitable for restart to active record.
PXP04RIS	04	1A	GET Restart to Active Record: 'Restart to active record' specified inconsistently together with positioning to line, page, or end of queue entry.
PXP04NRU	04	1B	GET Restart to Active Record: 'Restart to active record' request is rejected because requestor is not in browse mode.
PXP08SPL	08	01	Invalid SPL - VSE/POWER received an invalid SPL: - The SPL has no descriptor 'SPL' in the first three bytes. - There exits a length inconsistency concerning OPTBs: the offset specified in SPLEOPOF plus the length of OPTBs is greater than the SPL received, or the offset in field SPLEOPOF is wrong.
PXP08REQ	08	02	Unknown request type in SPL - The request indicated in request byte SPLGRQB is not PUT, GET, CTL, or GCM.

Table 80. Return and Feedback Codes and Their Meanings (continued)

Mnemonic PXPFBKCD	Return Code	Fdbk Code	Meaning
PXP08SRQ	08	03	Unknown subrequest type in SPL - The subrequest indicated in subrequest byte SPLGSRB is not valid. Valid subrequests are: - DISPLAY job/output - CANCEL job/output - RELEASE job/output - HOLD job/output - DELETE job/output - ALTER job/output - VSE/POWER command passed - Delete checkpoint information
PXP08FB2	08	04	Unknown function byte 2 in SPL - For an ALTER job/output subrequest indicated in SPLGSRB the function specified in SPLGFB2 is invalid. Valid functions are: - ALTER class - ALTER disposition - ALTER number of copies - ALTER compaction table name - ALTER remote id - ALTER priority - ALTER system id - ALTER destination node name - ALTER destination user id
PXP08JNM	08	05	Job name invalid or missing - The job name must be alphaj and not longer than eight characters.
PXP08QID	08	06	Queue identifier invalid or missing - The queue id passed in the SPL is not LST, PUN, RDR or XMT or the queue id is not allowed for the specified service.
PXP08CLS	08	07	Class specification invalid or missing - The specified class is invalid (not A-Z or 0-9).
PXP08PWD	08	08	Password invalid - The password must be alphaj and not longer than eight characters.
PXP08UID	08	09	User id invalid or missing - The user id is mandatory. It must be alphaj and not longer than eight characters.

Table 80. Return and Feedback Codes and Their Meanings (continued)

Mnemonic PXPFBKCD	Return Code	Fdbk Code	Meaning
PXP08RFM	08	0A	Record format invalid - The record format indicated in SPLORCFM is invalid. Valid record formats are: - SCS Print - BMS Mapping - 3270 format - CPDS format - Escape mode - ASA format - Machine Control format
PXP08DSP	08	0B	Disposition invalid - The local disposition specified in SPLDDP is not D, K, L or H, or the transmission disposition specified in SPLOTDP is not D, K, L, or H.
PXP08PRY	08	0C	Priority invalid - The priority specified in SPLDPR is not in the range from 1-9.
PXP08SID	08	0D	Sysid invalid - The sysid specified in SPLDSID is neither in the range from 1-9 nor 'N'.
PXP08TNN	08	0E	Destination node name invalid - The first character must be alphap, the rest alpham and it may not be longer than eight characters.
PXP08TUN	08	0F	Destination user id invalid - The user id must be alphaj* and not longer than eight characters.
PXP08FNO	08	10	Forms id invalid - The forms id must be alphaj and not longer than eight characters.
PXP08FCB	08	11	FCB name invalid - The first character must be alphap, the rest alpham and it may not be longer than eight characters.
PXP08UCB	08	12	UCB name invalid - The first character must be alphap, the rest alpham and it may not be longer than eight characters.

Table 80. Return and Feedback Codes and Their Meanings (continued)

Mnemonic PXPFBKCD	Return Code	Fdbk Code	Meaning
PXP08FLH	08	14	Flash id invalid - The flash id must be alphaj and not longer than four characters.
PXP08CPT	08	15	Compaction table name invalid - The first character must be alphap, the rest alpham and it may not be longer than four characters.
PXP08CGP	08	16	Copy group parameter invalid - The copy group must be numeric and the sum of the individual copy groups may not exceed 255. A maximum of eight bytes is allowed.
PXP08CHR	08	17	Character arrangement table invalid - The character arrangement table must be alphaj and not longer than four characters.
PXP08MOD	08	18	Copy modification name invalid - The copy modification name must be alphaj and not longer than four characters.
PXP08CCR	08	19	Character arrangement table for copy modification invalid - The table must be alphaj and not longer than four characters.
PXP08BTS	08	1A	Reply buffer too small - The reply buffer your program provides is too small to hold - the requested OPTBs - the requested order or signal - an SPL - a data record plus prefix - a message plus prefix - the extended checkpoint VSE/POWER returns the required buffer length in field PXPRBLN of the user data.
PXP08IAO	08	1B	Append/Restart specification invalid - Append or Restart is specified in function byte SPLGFB1, but Append or Restart is not allowed for RDR queue entries.

Table 80. Return and Feedback Codes and Their Meanings (continued)

Mnemonic PXPFBKCD	Return Code	Fdbk Code	Meaning
PXP08IAB	08	1C	Action byte invalid - PUT Service: A null buffer was sent to VSE/POWER but the action indicated in PXUACT1 is invalid. Valid actions are: - End of Data - Segment request - EOD for appendable output - Checkpoint request - Quit request Get Service: The action indicated in PXUACT1 is invalid. Valid actions are: - Quit request - Printing/punching failed - Close request - Purge request - Send data request - Flush hold request For a spool-access support service an action was indicated in PXUACT1 which is only valid for External Device Support.
PXP08ICR	08	1D	Control record received invalid - PUT Service: A control record is indicated in the user data, but the length of the control record is invalid. GET Service: A control record is indicated in the user data, but the type of the control record is invalid. Valid control records are: - Restart control record - Checkpoint control record - Get-OPTB control record - Modify-OPTB control record CTL Service: On request of a PDISPLAY command a temporary queue entry is built and passed to the user program. During passing the result of the command to the user program only a restart control record is allowed. There is a length mismatch between the length specified in the prefix and the actual length of the control record passed.
PXP08PRG	08	1E	Programmer name invalid - The programmer name must be alphajb and not longer than twenty characters.
PXP08ROO	08	1F	Room number invalid - The room number must be alphajb and not longer than eight characters.

Table 80. Return and Feedback Codes and Their Meanings (continued)

Mnemonic PXPFBKCD	Return Code	Fdbk Code	Meaning
PXP08DPT	08	20	Department number invalid - The department number must be alphajb and not longer than eight characters.
PXP08BLD	08	21	Building number invalid - The building number must be alphajb and not longer than eight bytes.
PXP08CON	08	22	Conflicting specifications For unique determination of the reason, see the additional feedback-2 code PXPFBKC2, described in "Spool-Access Support Parameter List (PWRSPL DSECT)" on page 231.
PXP08ROL	08	23	Received record too large - The data record received has a record length greater than 32K-8. VSE/POWER passes the offset from the beginning of user program's send buffer to the invalid record back in field PXPROFF of the user data.
PXP08IBT	08	24	Buffer type invalid - PUT Service: An update SPL was received for a PUT Service Job. Specified buffer type is invalid. Valid buffer types are: - SPL - normal data buffer - control record CTL Service: The buffer received is no control record.
PXP08ROS	08	25	Request out of sequence For unique determination of the reason, see the additional feedback-2 code PXPFBKC2, described in "Spool-Access Support Parameter List (PWRSPL DSECT)" on page 231.
PXP08SOS	08	26	SPL out of sequence - GET Service: VSE/POWER receives an SPL while processing a GET Service. PUT Service: VSE/POWER receives an SPL during normal PUT processing or VSE/POWER receives an update SPL with checkpoint or quit indication in PXUACT1.

Table 80. Return and Feedback Codes and Their Meanings (continued)

Mnemonic PXPFBKCD	Return Code	Fdbk Code	Meaning
PXP08BOS	08	27	There is currently no service in process, but VSE/POWER received a normal data buffer or a control record. Expected is a SPL.
PXP08RPH	08	28	Request prohibited - PUT Service: Segment, Checkpoint or Append is indicated in PXUACT1, but spooling is done to the RDR queue. GET Service: Flush hold is indicated in PXUACT1 but Flush hold is only allowed for External Device Support.
PXP08ISS	08	29	Signal specified invalid or out of sequence - VSE/POWER received an invalid signal. Valid signals are: - device stopped signal - setup-processed signal VSE/POWER received a setup-processed signal, but no SETUP order was sent to the DDS. VSE/POWER received a device stopped signal, but no STOP device order was sent to the DDS.
PXP08RPW	08	2A	Record prefix invalid - VSE/POWER received a buffer with a length which is less than the length of a record prefix plus 1 data byte. The remaining length of the buffer where VSE/POWER deblocks the data records from, is less than the length of the record prefix plus 1 data byte. For spooling data to RDR or PUN queue, normal data record must be specified in the record prefix. For spooling data to the LST queue normal data record or CPDS record must be specified in the record prefix. Record format APA was specified in SPLORCFM, but in the record prefix CPDS data record is not specified. VSE/POWER passes the offset from the beginning of user program's send buffer to the invalid record back in field PXPROFF of the user data.

Table 80. Return and Feedback Codes and Their Meanings (continued)

Mnemonic PXPFBKCD	Return Code	Fdbk Code	Meaning
PXP08FB1	08	2B	Unknown function byte 1 - PUT Service: The specified function in SPLGFB1 is invalid for PUT Service. Valid functions are: - Append of incomplete queue entry - Restart of queue entry GET Service: The specified function in SPLGFB1 is invalid for GET Service. Valid functions are: - Browsing of queue entry - Generic get request
PXP08IML	08	2C	Maximum record length invalid - The maximum record length specified in SPLDLREC is for PUT Job 128 bytes and for PUT Output 32K-8.
PXP08IEX	08	2D	Subsystem name invalid - The subsystem name must be alphaj* and not longer than eight characters.
PXP08SPA	08	2E	Spanned record received - The record length specified in the prefix indicates that the data goes beyond the buffer end. VSE/POWER passes the offset from the beginning of user program's send buffer to the invalid record back in field PXPROFF of the user data.
PXP08ICC	08	2F	Carriage control character invalid - The record prefix contains a carriage control character of X'FF', X'FD' or X'FE'. This is invalid. VSE/POWER passes the offset from the beginning of user program's send buffer to the invalid record back in field PXPROFF of the user data.
PXP08IOR	08	30	Inbound order invalid - The order request byte contains an invalid order. Valid orders are: - Send message order - Set logical destination order - Put account record order
PXP08JNO	08	31	Invalid job number- The job number is X'00' or, within a CTL service, the job suffix number is not 0 but the job number is 0.

Table 80. Return and Feedback Codes and Their Meanings (continued)

Mnemonic PXPFBKCD	Return Code	Fdbk Code	Meaning
PXP08JSF	08	32	Job suffix invalid - The suffix specified in SPLGJS is greater than 127.
PXP08IUI	08	33	User information invalid - The user information must be alhpaj and not longer than 16 bytes.
PXP08IPD	08	34	Service invalid for a DDS - The following services are not allowed for a DDS: - PUT Service - GET Service to the RDR/XMT queue
PXP08UXR	08	35	Response received is invalid - The message length of the order response is larger than the buffer received by VSE/POWER. The buffer received contains only the order response record header.
PXP08WOS	08	36	'Wait for order' request out of sequence - Currently not used!
PXP08NSP	08	37	Separator pages/cards invalid - The number of separator pages/cards must be in the range from 1-9.
PXP08IRR	08	38	Control request invalid - PUT Service: An unexpected or unknown control record type has been found, or one of the following control requests is not allowed for the RDR queue: - Restart control record - Get-OPTB control record - Modify-OPTB control record GET Service: The following requests are not allowed for the RDR queue: - Get-OPTB control record - Modify-OPTB control record
PXP08IOP	08	39	OPTB invalid - The format or contents of an OPTB is invalid.

Table 80. Return and Feedback Codes and Their Meanings (continued)

Mnemonic PXPFBKCD	Return Code	Fdbk Code	Meaning
PXP08OLM	08	3A	OPTB length mismatch - Modify-OPTB: there is a length mismatch between the current OPTB in the DSHR and the modification.
PXP08DOP	08	3B	Duplicate OPTBs - VSE/POWER received an SPL for PUT Output service, but there are duplicate OPTBs specified.
PXP08OTL	08	3C	Specified OPTBs too long - The OPTBs specified in the SPL exceed the maximum length of 32760 for a total DSHR.
PXP08IDH	08	3D	Invalid DSHR found - Get-OPTB, Modify-OPTB: While reading an OPTB in the DSHR the following inconsistencies are found: - the count of the number of data elements supplied for the keyword parameter is greater than 16,383. - the length of one data element is greater than 16,383. - the length of all data elements supplied for the keyword parameter is greater than the OPTB section of DSHR.
PXP08DIS	08	3E	Distribution code invalid - The distribution code must be alphaj.
PXP08INK	08	3F	Invalid keyword OPTB - The syntax of the keyword OPTB is wrong. Thus, VSE/POWER could not find the keyword delimiter '=', or the keyword OPTB starts with '='. VSE/POWER returns the offset from the beginning of the user programs' send buffer to the invalid OPTB in field PXPROFF of the user data.
PXP08NDK	08	40	No define statement present - There was no DEFINE statement specified for the passed keyword OPTB during autostart of VSE/POWER. VSE/POWER returns the offset from the beginning of the user programs' send buffer to the invalid OPTB in field PXPROFF of the user data.

Table 80. Return and Feedback Codes and Their Meanings (continued)

Mnemonic PXPFBKCD	Return Code	Fdbk Code	Meaning
PXP08IDV	08	41	Invalid keyword OPTB value - The keyword OPTB value passed does not meet the definitions of the respective DEFINE statement or has a wrong syntax. VSE/POWER returns the offset from the beginning of the user programs' send buffer to the invalid OPTB in field PXPROFF of the user data.
PXP08CKZ	08	42	Length of extended checkpoint information is zero.- A checkpoint record has been received indicating that extended information exists, but this extended information has a length of zero.
PXP08CKL	08	43	Length of extended checkpoint information is larger than the currently used DBLK size minus 288.- A checkpoint record has been received indicating that extended information exists, but this extended information has an invalid length.
PXP08IQN	08	44	Internal queue-entry number invalid- The number is either zero or too large for the current - 'Delete checkpoint information' request or - 'Direct' PDISPLAY CTL request or - 'Direct' PALTER/PDELETE/PHOLD/PRELEASE CTL request or - 'Direct' GET-Service request.
PXP08GJN	08	45	Generic jobname can not be used-... A 'delete checkpoint information', or a 'Direct' PALTER/PDELETE/ PHOLD/PRELEASE CTL request, or a 'Direct' PDISPLAY CTL request has been issued, which can only address a single, unique queue entry.
PXP08SEU	08	46	VSE security userid is invalid- A security password was specified, but the security userid was omitted.
PXP08SEP	08	47	VSE security password is invalid- Either a security password was specified, but the length exceeds the maximum allowed by VSE, or an authorized program has not specified both security userid and password when the VSE Access Control was not active, or a security userid was specified and the password was required but omitted. Blank padding to the right determines the password length.

Table 80. Return and Feedback Codes and Their Meanings (continued)

Mnemonic PXPFBKCD	Return Code	Fdbk Code	Meaning
PXP08IPM	08	48	Incorrect processing mode for PUT OPEN OUTPUT - The SPLXPMDE field does either not contain 8 blank characters (default), or does not contain up to 8 alphaj characters, leftbound and padded with blanks to the right. For suitable processing mode values, refer to the PRMODE=operand of the VSE/POWER * $$ LST/PUN statement in *VSE/POWER Administration and Operation*.
PXP08IEM	08	49	PUT service: invalid expiration value specified in the SPLXEXPD field, or in the SPLXEXPH field of SPL, or in both (SPLXEXPD must be a number between 0 and 999, and SPLXEXPH must be a number between 1 and 24).
PXP08SDU	08	4A	GET service: Modify-OPTB request is rejected because the spooled master and duplicate queue entries must not be changed.
PXP08RDU	08	4B	PUT-output service: PUT-OPEN-RESTART request is rejected because the spooled master and duplicate queue entries must not be changed.
PXP08XUA	08	4C	GCM-XEM service: GCM-XEM-START request is rejected because service is unavailable or cannot be started for application. For unambiguous identification of the reason, see the additional feedback-2 code PXPFBKC2, described in "Spool-Access Support Parameter List (PWRSPL DSECT)" on page 231.
PXP0CINS	0C	01	User program used SEND instead of SENDR.
PXP0CIXF	0C	02	User program used CLEAR.
PXP0CBTL	0C	03	Buffer too large - The buffer received by VSE/POWER is greater than 64KB.
PXP0CPER	0C	04	Protocol error - VSE/POWER received a buffer and assumes an order response control record, but the buffer contains at least the header of an order or it is neither an order response nor another order type.
PXP0CPVD	0C	05	Protocol violated by a DDS - Currently not used!
		06	Currently not used!
PXP0CIOE	0C	07	I/O error occurred on queue/data/account file -

Table 80. Return and Feedback Codes and Their Meanings (continued)

Mnemonic PXPFBKCD	Return Code	Fdbk Code	Meaning
PXP0CSNF	0C	08	VSE/POWER section not found in job (JHR) or data set header record (DSHR) - may be due to data corruption.
PXP0CCOR	0C	09	Job or data set header record (DSHR) is corrupted (some length field is either zero or greater than X'7FFF')
PXP10CAA	10	03	Connection already active - ICCF or DSNX tried to establish a second notify path. CICS notify user already connected. GCM: A wait request is already connected.
PXP10PSP	10	05	PSTOP given by operator or due to exit failure or stop by exit
PXP10SIE	10	06	Severe internal error
PXP10MST	10	07	The number of Spool Access Support (SAS) GET/PUT/CTL/GCM tasks has reached the VSE/POWER limit of concurrent tasks as reported on console by message 1Q3JA. Therefore the CONNECT attempt to SYSPWR had to be rejected by an XPCC DISCPRG request. The default limit of 250 tasks may be modified by the 'PVARY MAXSAS,nnnn' command.

Part 3. Exit Routines

Chapter 15. Writing Various Exit Routines

VSE/POWER supports user-written routine types for the customized handling of local input (JOBEXIT), input from the network (NETEXIT), output to the network (XMTEXIT), and for output (OUTEXIT).

All user routines are loaded with the POWER generation macro or the PLOAD command and disabled or enabled with the PVARY command. To display status information about the exit routine, use the PDISPLAY EXIT command. For details on the POWER generation macro and the PLOAD, PVARY, and PDISPLAY commands, see *VSE/POWER Administration and Operation*, SC34-2625. For a discussion of the routines for network input/output, see *VSE/POWER Networking*, SC34-2603.

Intercommunication Between Exit Routines

When exits gain control, the TCB field TCQV (use mapping macro IPW$DTC) points to the queue record (use macro IPW$DQR) of the queue entry being processed. The one-byte field QRUEX of the queue record has been provided by VSE/POWER for exit purposes only. VSE/POWER neither checks nor processes field QRUEX. Exits may use it, for example, as follows:

The NETEXIT can set one or more flags in QRUEX while receiving a job. When the job starts to execute in a z/VSE partition, all of its created output queue entries will inherit the contents of QRUEX. An XMTEXIT or OUTEXIT can then check QRUEX of the created output entries and take appropriate action. This is an easy way to pass information from an input exit routine like JOBEXIT/NETEXIT to output exit routines like OUTEXIT/XMTEXIT. The contents of QRUEX are lost when the queue entry is transmitted to another node.

Handling of Exit Failures

When a failure has been detected in an exit routine, recovery will be performed if the following conditions are fulfilled:

- The exit gets control in the normal way, that is, the exit is called properly by the VSE/POWER task in the exit calling module. This will set the task in 'exit-state' when the exit is called.
- The contents of registers 10 and 11 are not destroyed by the exit, which means that R11 must address the task control block of the task which uses the exit, and R10 must address the Common Address Table.

When VSE/POWER performs recovery from such a failure, message 1Q2CI is issued reflecting the exit failure and then message 1Q2KI to indicate continuation of VSE/POWER processing. VSE/POWER produces an IDUMP for better locating the error. If IDUMP processing fails, VSE/POWER does not ask for a device to print the dump. Tasks which are currently using the exit are stopped immediately. Tasks which may use the failing exit, but are currently not processing a queue entry, are not stopped immediately, but at the time they start processing and are going to call the exit. The exit remains enabled, but is marked with the indicator 'failed'. VSE/POWER issues message 1Q2HI which informs the operator that the exit has been put into 'FAILED STATE'. Other task types which do not use the faulty exit continue processing.

The report issued in response to a PDISPLAY EXIT command reflects the exit with state 'failed'.

Recovery will not be performed if the task is **not** in exit-state when the failure occurs.

However, if the failing exit destroyed data vital for further VSE/POWER processing and recovery has been done, unpredictable results may occur. For example, VSE/POWER may terminate abnormally after a while although failure recovery has been performed previously. No checkpointing of VSE/POWER control blocks takes place.

Whenever an exit is in 'FAILED STATE', a newly started task is stopped at the time the task gives control to the exit. Sometimes, especially if a JOBEXIT failed, it must be possible to start a task which should not be stopped. This can be achieved by one of the following actions:
- Put the user exit into 'DISABLED STATE' by using the PVARY command with the DISAB operand.
- Load a new user exit by using the PLOAD command.

Recovery Feasible

Recovery takes place in several steps. VSE/POWER
1. Issues message 1Q2CI indicating the failing exit,
2. Issues message 1Q2KI which signals that recovery is performed,
3. Takes an IDUMP for locating the error,
4. Marks the failing exit as 'FAILED',
5. Issues message 1Q2HI,
6. Forces all tasks which are currently using the exit to stop immediately,
7. Returns to the location from where the exit was called,
8. Issues appropriate messages for each task which is stopped (for example, 1Q33I, 1QX3I, 1QY4I, 1RA8I),
9. Allows other tasks to continue processing.

Resulting Final Task Processing

If recovery is performed, all tasks which use the exit are stopped. Depending on the task which actually terminates, an additional message is issued reflecting the task type and exit failure condition:

1Q33I Local reader or writer task is stopped (equivalent to PSTOP command).

1QX3I Cross-partition task which spooled a job to the reader queue is stopped (equivalent to PSTOP command).

1QY4I Device service task (using the spool-access support) is stopped (equivalent to PSTOP DEV command).

1RA8I Network receiver or transmitter task is drained. Behavior is equivalent to

`PDRAIN PNET,nodeid,RV*|TR*,JOB` and
`PDRAIN PNET,nodeid,RV*|TR*,OUT`, respectively.

Since all tasks which use the faulty exit are stopped, such a message may occur repeatedly.

Note:

1. If a local reader task is stopped, part of the input job remains in the reader. Running under VM, the size of this remaining part varies depending on the buffersize used by VM. For small jobs, the size is usually zero. For large jobs, the size is usually not zero. The first statement remaining in the reader is usually not that statement which follows immediately the last statement processed by VSE/POWER.

   To remove the remaining part, you can issue the VM command CLOSE cuu, where 'cuu' is the device address of your reader.
2. No messages appear if a reader or writer task for a remote work station (using either BSC or SNA protocol) is stopped.

Handling of Data being Processed by Stopped Tasks

Depending on the type of exit, VSE/POWER handles incoming or outgoing data as follows when recovery is performed due to a faulty exit:

JOBEXIT
The incoming VSE/POWER job is not added to the queue.

NETEXIT
Incoming data is not added to queue.

OUTEXIT
Outgoing data is queued again with its original disposition.

XMTEXIT
Outgoing data is queued again with its original disposition.

Recovery Not Feasible

When the failure occurred in an exit but recovery cannot be performed:

1. Message 1Q2CI or 1Q2DI, respectively, is issued, reflecting the failing exit, and
2. VSE/POWER termination takes place.

Exit Routine for Local Input (Type JOBEXIT)

Function

The routine gets control from VSE/POWER when a z/VSE JCL or a VSE/POWER JECL statement is read, unless DISP=I was specified. The JCL or JECL statement may be read from a local or remote reader or from a programmed interface for job submission via the spool-access PUT request or the PUTSPOOL request. VSE/POWER passes to the routine all statements of the types listed below, including continuation lines, if there are any:

All statements beginning with //

All statements beginning with /.

All statements beginning with *

/*

/&

and has converted all VSE/POWER JECL statements and the z/VSE// JOB JCL statements to upper case characters.

The routine may change or delete any z/VSE job control or JECL statement; it may insert other statements.

If the exit routine inserts a statement, VSE/POWER handles it and then presents to the routine once more the original statement. When the exit routine has made all the insertions it must indicate if it wants to delete or to process the original statement.

Note: If the routine inserts statements after it has passed the * $$ EOJ statement, VSE/POWER runs out of processor storage during end-of-job processing, and the operator gets a wait message. This should be avoided.

The User Routine Work Area

The routine must be re-enterable if several VSE/POWER reader tasks are running at the same time. A reader task in this context is:

- A task started by a PSTART RDR,... command
- A task servicing a job-input spool request from another partition
- An RJE reader task started when a terminal sends job data to VSE/POWER

To help you to make this input routine reentrant, VSE/POWER passes a work area to the routine. This work area can be used by the routine to hold variables and information needed with the queue entry or to hold records to be inserted. You define the size of the work area in the POWER macro or with the PLOAD command (maximum 65,535 bytes, default zero) along with the phase name of your exit routine under the parameter JOBEXIT (no longer RDREXIT).

The work area is obtained from the partition GETVIS area and is initialized with X'00'. The first four bytes of the work area contain its length. This information is refreshed every time the user routine gets control. It exists as long as the reader task exists and is **not initialized with every new job as for NETEXIT, XMTEXIT, and OUTEXIT type routines.**

Restrictions

The routine may not perform an operation that causes a wait condition and may not use any z/VSE macro that needs the logical transient area (LTA); the DUMP macro may, for example, cause a deadlock condition.

Note: The exit routine must **not** use access-register mode and all executable code must be located below the 16MB line. The exit routine is executed in 24-bit addressing mode and must call VSE/POWER services and return to VSE/POWER in 24-bit addressing mode.

When the exit routine loses control (for example, due to a page fault), the status of the currently used addressing mode and the access registers are not saved. Thus, when the exit routine gets control back again, the previously used access registers can not be restored.

Because the exit routine must not use the access-register mode, the exit routine can not use data spaces. VSE/POWER does not accept an exit routine which has already been loaded into the SVA.

Interface Description

Use of Registers

When the user routine gets control, this is the register content:

```
Register 0:          contains the address of the statement read
Register 1:          contains the length of the statement
Register 3:          points to the generated work area, otherwise zero if none
----------
Registers 10 to 12:  reserved for VSE/POWER and may not be changed.
                     Register 11: points to the TCB of the reader task
                     and may be used to identify it.
Register 14:         return address to VSE/POWER.
Register 15:         entry point to user routine.  When leaving the user
                     exit routine, to be used for its return code.
```

To return to VSE/POWER, issue a BR 14 instruction.

Return Codes

The user routine must return control to VSE/POWER with one of the following return codes in register 15:

X'00' Process the statement passed to the routine. The user routine may update fields within the statement but may not change its length or address.

X'04' Ignore this statement.

X'08' Insert and process the new statement; return the original statement to the user routine once more. Any number of statements may be inserted.

The address of the statement that is to be inserted must be provided in register 0, the statement's length in register 1. This length must be X'50'.

Note: Inserted VSE/POWER JECL and z/VSE // JOB JECL statements will be translated to upper case before spooling.

X'0C' Terminate the z/VSE job — that means no further statements of the current z/VSE job unit (starting with a // JOB, ending with the next /& or // JOB statement) should be spooled for creating a VSE/POWER reader queue entry. Also no more JCL statements (//, /., /*, /&) of the current z/VSE job unit are passed to the jobexit. It is recommended to request X'0C' only, when a // JOB statement has been passed to the jobexit, otherwise already spooled statements of the z/VSE job unit may create a job fragment, which VSE/POWER terminates with a /& statement.

X'10' Terminate and flush the VSE/POWER job — that means no further statements of the current VSE/POWER job unit (starting with a * $$ JOB, ending with the next * $$ EOJ or a * $$ JOB statement) should be spooled for creating a VSE/POWER reader queue entry, and the whole VSE/POWER job fragment should not be added to the reader queue at all. Also no JCL statements (//, /., /*, /&) and potential JECL statements (starting with *) of the current VSE/POWER job unit are passed to the jobexit.

If, at the first statement of a VSE/POWER job, there exists a return code of X'0C' or X'10', VSE/POWER issues a message and continues processing.

If the register 15 return code does not show one of the before mentioned settings, VSE/POWER issues message 1R57I and terminates the VSE/POWER job, as if return code X'10' was given.

If ACCOUNT support is available, the field RDRNUM in the reader account record (see Table 9 on page 21) reflects records added or deleted by the user routine.

Tracing of Exit Failures

When debugging logic failures of your exit routine, it may be helpful to obtain a snapshot dump of the VSE/POWER partition at a predefined processing point of your routine. You should include the call of macro IPW$IDM into the exit routine to have an Idump generated in flight. For details, refer to *VSE/POWER Administration and Operation*, SC34-2625.

JOBEXIT Programming Example

The sample of the JOBEXIT routine shown here is delivered to you as an A-book in PRD1.MACLIB under the name of JOBEXAMP.A. If you extend this example further, you may need macros that are only available with the optional code material.

The sample includes:

1. The set of statements that causes the source code of the JOBEXIT routine to be assembled and cataloged;
2. The source code of the routine example itself.

Control Statements to Assemble and Catalog the Routine

```
* $$ JOB JNM=JOBEXRUN,CLASS=A,DISP=D
// JOB JOBEXRUN
// OPTION CATAL
// LIBDEF *,SEARCH=PRD1.MACLIB
// LIBDEF *,CATALOG=...
*
* PROVIDE FOR ... CATALOG SUBLIBRARY FOR JOBEXAMP
*
// EXEC ASSEMBLY,SIZE=100K
         COPY JOBEXAMP
         END
/*
// EXEC LNKEDT
/&
* $$ EOJ
```

Programming Example Source Code

```
         TITLE ' JOBEXAMP'
         PUNCH ' PHASE JOBEXAMP,*'
         SPACE
*******************************************************************
***                                                             ***
***                 J O B E X A M P                             ***
***                                                             ***
***           VSE/POWER JOB EXIT: EXAMPLE PROGRAM     DY45811  ***
***                                                             ***
*******************************************************************
         SPACE 2
*        THIS PROGRAM - NAMED JOBEXAMP - ACTS AS AN EXAMPLE FOR A
*        USER WRITTEN JOB EXIT ROUTINE.
         SPACE 2
*        THIS EXAMPLE GIVES A SKELETON WHICH SHOWS HOW TO CHECK
*        THE CONTENTS OF JCL AND JECL STATEMENTS. DEPENDING ON THE
*        NEEDS OF THE CUSTOMERS AN ACTION TYPE MAY BE PROVIDED
*        BY THE JOB EXIT TO POSITION 80 OF THE STATEMENT.
*        THIS ACTION TYPE WILL BE INTERPRETED LATER ON AND THE
*        APPROPRIATE RETURN CODE WILL BE
*        SET IN REGISTER 15 TO BE PASSED TO VSE/POWER.
```

```
         SPACE 2
*        CONTROL IS GIVEN TO THIS JOB EXIT VIA REGISTER 15
*        BY THE LOGICAL READER (IPW$$LR).
         SPACE 2
*        THIS JOB EXIT IS ASSUMED TO BE LOADED WITH A WORK AREA
*        OF 50 BYTES.
         SPACE 2
*        THE FOLLOWING ADDRESSABILITY IS ASSUMED AT ENTRY TO
*        THE JOB EXIT:
         SPACE
*        R0  -  ADDRESS OF STATEMENT PASSED BY VSE/POWER
*        R1  -  LENGTH  OF STATEMENT PASSED BY VSE/POWER
*        R3  -  ADDRESS OF THE WORKAREA PASSED BY VSE/POWER
*        RA  -  ADDRESS OF VSE/POWER COMMON ADDRESS TABLE (CAT)
*        RB  -  ADDRESS OF TASK CONTROL BLOCK (TCB)
*        RE  -  RETURN ADDRESS TO VSE/POWER
*        RF  -  BASE REGISTER OF THIS ROUTINE
         SPACE 2
*        THIS JOB EXIT PHASE IS LOCATED WITHIN THE
*           - PAGEABLE AREA OF THE VSE/POWER PARTITION WHEN LOADED
*             AT VSE/POWER INITIALIZATION TIME, OR IN THE
*           - GETVIS AREA OF THE VSE/POWER PARTITION WHEN LOADED
*             AFTER INITIALIZATION  BY THE 'PLOAD' COMMAND.
*        FOR DEBUGGING LOCATE THE JOB EXIT IN STORAGE
*        BY ITS STORAGE DESCRIPTOR 'JOB-EXIT' AND BY THE POINTER
*        'CARE' OF THE COMMON ADDRESS TABLE OF
*        VSE/POWER.
         SPACE 2
*        THE FOLLOWING PIECE OF CODE IS ONLY AN EXAMPLE.
*        IT IS THE USER'S RESPONSIBILITY TO DEVELOP
*        HIS OWN ROUTINE CONCERNING HIS PROBLEM DEFINITION.
         EJECT
*        REGISTER USAGE:
*           R0 - ADDRESS OF STATEMENT IN CASE OF INSERT
*           R1 - LENGTH OF STATEMENT IN CASE OF INSERT
*           R2 - ADDRESS OF STATEMENT IN JOB EXIT
*           R3 - ADDRESS OF WORK AREA PASSED BY VSE/POWER
*           R4 - R7  -  WORK REGISTER
*           R8 - RETURN CODE SET BY JOB EXIT
*           R9 - WORK REGISTER
*           RA - RC USED BY VSE/POWER
*           RE - RETURN ADDRESS TO VSE/POWER
*           RF - EXIT BASE REGISTER AND RETURN CODE TO VSE/POWER
         SPACE 2
JOBEXAMP CSECT 0                   ESTABLISH MAIN CONTROL SECTION
         USING *,RF                BASE REG ESTABLISHED BY VSE/POWER
         USING PADS,RA             ADDRESSABILITY OF VSE/POWER CAT
         USING TCDS,RB             ADDRESSABILITY OF TCB
         USING CDSECT,R2           ADDRESSABILITY OF STATEMENT
         USING WDSECT,R3           ADDRESSABILITY FOR WORK AREA
         B     JEX000              SKIP STORAGE DESCRIPTOR
         SPACE
         DC    CL12'JOB-EXIT'      DEFINE STORAGE DESCRIPTOR
         SPACE
*******************************************************************
*                                                                 *
*        THE FOLLOWING PIECE OF CODE IS USED TO CHECK THE         *
*        STATEMENT. FOR A * $$ JOB STATEMENT THE ACTION TYPE      *
*        WILL BE DETERMINED.                                      *
*        FOR ALL OTHER STATEMENTS CONTROL IS PASSED TO VSE/POWER  *
*        WITH NORMAL PROCESSING INDICATED IN REGISTER 15.         *
*                                                                 *
*        FOR A JOB STATEMENT ACCEPTED BY THE JOB EXIT             *
*        A NEW STATEMENT WILL BE INSERTED AND THE JOB COUNTER     *
*        IN THE WORK AREA WILL BE INCREMENTED.                    *
*        INSERTION OF THE STATEMENT IS INDICATED IN THE WORK      *
*        AREA TO AVOID LOOPING SINCE THE CURRENT * $$ JOB         *
```

```
*        STATEMENT IS PASSED AGAIN TO THIS JOB EXIT.                *
*                                                                   *
*        FIRST THE EXAMPLE CHECKS,WHETHER THE REQUIRED WORKAREA     *
*        SIZE OF 50 BYTES HAS BEEN SPECIFIED FOR THIS EXIT.         *
*        IF NOT, WE DO NOT DARE TO USE THE PASSED WORKAREA,         *
*        INSTEAD THE CENTRAL OPERATOR WILL BE INFORMED BY A         *
*        WARNING MESSAGE.                                           *
*                                                                   *
*********************************************************************
         SPACE
JEX000   DS    0H
         SPACE 2
*        WHENEVER A WORK AREA IS USED, THE LENGTH OF THIS AREA MUST BE
*        VERIFIED EACH TIME THE EXIT GAINS CONTROL, BECAUSE
*        - THE EXIT MIGHT HAVE BEEN LOADED BY OPERATOR WITH A WRONG
*          LENGTH OF ITS WORK AREA
*        - A NEW VERSION OF THE EXIT MIGHT HAVE BEEN LOADED AND THE
*          PREVIOUSLY DEFINED WORK AREA IS STILL USED UNTIL THE TASK
*          ENDS.
         SPACE 2
*        SINCE A WORK AREA IS REQUIRED BY THIS ROUTINE, VERIFY IF ONE
*        IS SPECIFIED. IF NOT, THE VSE/POWER JOB IS FLUSHED.
         SPACE 1
         LTR   R3,R3                   WORK AREA SPECIFIED ?
         BZ    JEX003                  ..NO, WARN OPER. AND FLUSH JOB
         SPACE 2
*        THE WORK AREA EXISTS AND CAN NOW BE ADDRESSED. TEST IF
*        THE WORK AREA IS LARGE ENOUGH.
         SPACE 1
         L     R4,WDLENGTH             GET SPECIFIED WORK AREA SIZE
         LA    R5,WDLEN                GET REQU. SIZE OF WORK AREA
         CR    R4,R5                   WORK AREA TOO SHORT ?
         BNL   JEX005                  ..NO, CONTINUE EXIT
         SPACE 1
JEX003   DS    0H
         SPACE 1
*        RESPECT THAT MACRO IPW$WTO DESTROYS REGISTER R0-R3
         SPACE 1
         LA    R7,JEXMSG1          GET ADDRESS OF JEXMSG1
         BAL   R6,SUBWTO           ISSUE TEXMSG1
         SPACE 1
         LA    R7,JEXMSG2          GET ADDRESS OF JEXMSG2
         BAL   R6,SUBWTO           ISSUE TEXMSG2
         SPACE 1
         LA    R7,JEXMSG3          GET ADDRESS OF JEXMSG3
         BAL   R6,SUBWTO           ISSUE TEXMSG3
         SPACE 1
         LA    RF,X'10'            PROVIDE RETURN CODE TO FLUSH JOB
         BR    RE                  AND RETURN TO VSE/POWER
         SPACE 2
JEX005   DS    0H
         LR    R2,R0               SET UP ADDRESS OF STATEMENT
         CLC   COMPARE,CUSTNEED    * $$ JOB STATEMENT PASSED ?
         BE    JEX010              .. YES, CONTINUE
         B     JEX050              .. NO, CONTINUE
         SPACE
JEX010   DS    0H
         TM    WDFLAG,WDFLINS      INSERT OF STATEMENT DONE ?
         BZ    JEX020              ..NO, CONTINUE
         L     R4,WDCOUNT          GET CURRENT JOB COUNTER
         LA    R4,1(R4)            INCREMENT JOB COUNTER
         ST    R4,WDCOUNT          SAVE NEW VALUE
         NI    WDFLAG,X'FF'-WDFLINS RESET INSERT DONE
         B     JEX060              CONTINUE
         SPACE
JEX020   DS    0H
         SPACE
```

```
*******************************************************************
*                                                                 *
*        INSERT CODE HERE WHICH DETERMINES THE ACTION FOR         *
*        THIS JECL STATEMENT AND WHICH SAVES THIS ACTION          *
*        IN THE ACTION BYTE OF THE WORK AREA.                     *
*                                                                 *
*******************************************************************
         SPACE 4
*******************************************************************
*                                                                 *
*        THE FOLLOWING PIECE OF CODE IS USED TO REACT UPON THE    *
*        ACTION AND TO GET THE RELATED RETURN CODE FOR            *
*        VSE/POWER.                                               *
*        FOR AN ACTION 'FLUSH VSE/POWER JOB' THE APPROPRIATE      *
*        BIT IS SET IN THE FLAG BYTE OF THE WORK AREA.            *
*        THIS ACTION IS ONLY ACCEPTED BY VSE/POWER AFTER          *
*        PROCESSING OF THE * $$ JOB STATEMENT BY VSE/POWER.       *
*        SINCE THE JOB EXIT IS CURRENTLY PROCESSING THE * $$ JOB  *
*        STATEMENT FLUSH IS INDICATED WITH THE NEXT               *
*        STATEMENT.                                               *
*                                                                 *
*******************************************************************
         SPACE
JEX030   DS    0H
         LA    R8,4                ASSUME DELETE
         TM    WDACTION,WDACDEL    DO WE WANT TO DELETE THIS
*                                  STATEMENT ?
         BO    JEXDONE             IF YES, RETURN TO VSE/POWER
         SPACE
         TM    WDACTION,WDACINS    DO WE WANT TO INSERT ?
         BZ    JEX040              ... BRANCH IF NOT
         LA    R8,8                GET PROPER RETURN CODE
         LA    R0,INSERT           POINT TO CORRECT STATEMENT
         LA    R1,L'INSERT         GET PROPER LENGTH
         OI    WDFLAG,WDFLINS      INDICATE INSERT DONE
         B     JEXDONE             RETURN TO VSE/POWER
         SPACE
JEX040   DS    0H
         OI    WDFLAG,WDFLPOW      ASSUME FLUSH OF VSE/POWER JOB
         TM    WDACTION,WDACFLH    DO WE WANT TO FLUSH VSE/POWER JOB ?
         BO    JEX060              ..YES, CONTINUE
         SPACE
         NI    WDFLAG,X'FF'-WDFLPOW RESET FLUSH INDICATION
         SPACE
         TM    WDACTION,WDACCHG    DO WE WANT TO CHANGE THIS
*                                  STATEMENT ?
         BZ    JEX060              ..NO, CONTINUE
*        MVC   FIELD,NOTCHA        MOVE IN CHANGE INFO
         B     JEX060              CONTINUE
         SPACE
JEX050   DS    0H
         TM    WDFLAG,WDFLPOW      FLUSH VSE/POWER JOB TO DO ?
         BZ    JEX060              ..NO, CONTINUE
         LA    R8,16               INDICATE FLUSH VSE/POWER JOB
         NI    WDFLAG,X'FF'-WDFLPOW RESET FLUSH VSE/POWER JOB
         B     JEXDONE
         SPACE
JEX060   DS    0H
         SR    R8,R8               GET NORMAL RETURN CODE
         SPACE 1
*
*        R E T U R N   T O   V S E / P O W E R
*
         SPACE
JEXDONE  DS    0H
         MVI   WDACTION,X'00'      CLEAR ACTION BYTE
         LR    RF,R8               GET RETURN CODE
```

```
         BR    RE                  AND RETURN TO VSE/POWER LOG.RDR
         EJECT
***********************************************************************
*        SUBROUTINE TO ISSUE MESSAGE ON OPERATOR CONSOLE              *
*                                                                     *
* USAGE OF VSE/POWER MACRO IPW$WTO:                                   *
*       THIS MACRO, TOGETHER WITH SOME INDICATIONS SET IN THE TCB,    *
*       CAN BE USED TO DISPLAY A MESSAGE ON THE CENTRAL OPERATOR      *
*       CONSOLE.                                                      *
*       THE USAGE OF THIS MACRO DESTROYS THE REGISTER R0 - R3.        *
*       THE MESSAGE TO BE DISPLAYED SHOULD CONTAIN ALPHAMERIC         *
*       CHARACTERS ONLY.                                              *
*                                                                     *
***********************************************************************
         SPACE 2
SUBWTO   DS    0H
         STCM  R7,7,TCMW+1             PASS MESSAGE ADDRESS
         MVI   TCMW,X'00'              CLEAR OPTION BYTE
         OI    TCMW,TCPCOP             FORCE MESSAGE TO CONSOLE
         OI    TCMW,TCDNMM             SUPPRESS MESSAGE MODIFICATION
         SPACE
         IPW$WTO TCMW                  ISSUE MESSAGE
         SPACE
         MVI   TCMW,X'00'              CLEAR OPTION BYTE
         BR    R6                      RETURN TO CALLER
         SPACE 5
         DROP  R2,R3               RELEASE ADDRESSABILITY
         DROP  RA,RB               RELEASE ADDRESSABILITY
         EJECT
         SPACE
*******************************************************************
*        D E F I N I T I O N S                                    *
*******************************************************************
         SPACE
CUSTNEED DC    CL9'* $$ JOB '      ACTION FOR JOB STATEMENT REQUIRED
INSERT   DC    CL80'* THIS RECORD IS INSERTED BY JOB EXIT'
NOTCHA   DC    C'CHANGED'          CHANGE INFO
         SPACE 2
JEXMSG1  DC    AL1(JEXMSG1L-JEXMSG1-1)   LENGTH OF MESSAGE
         DC    C'INCORRECT SIZE OF WORKAREA GIVEN, 50 BYTES NEEDED'
JEXMSG1L EQU   *
         SPACE 1
JEXMSG2  DC    AL1(JEXMSG2L-JEXMSG2-1)   LENGTH OF MESSAGE
         DC    C'DISABLE JOBEXIT USING PVARY'
JEXMSG2L EQU   *
         SPACE 1
JEXMSG3  DC    AL1(JEXMSG3L-JEXMSG3-1)   LENGTH OF MESSAGE
         DC    C'STOP AND RESTART READER TASK. RELOAD JOBEXIT'
JEXMSG3L EQU   *
         SPACE 2
CDSECT   DSECT                     DSECT FOR JECL SATEMENT
COMPARE  DS    CL9                 PREFIX OF JECL STATEMENT
         DS    CL71                FILLER
         SPACE
WDSECT   DSECT ,                   DSECT FOR WORK AREA
WDLENGTH DS    F                   LENGTH OF WORK AREA
WDCOUNT  DS    F                   COUNTER TO MAINTAIN NUMBER OF
*                                  JOBS PROCESSED
WDACTION DS    X'00'               ACTION BYTE
WDACINS  EQU   X'80'               .. INSERT STATEMENT
WDACDEL  EQU   X'40'               .. DELETE STATEMENT
WDACFLH  EQU   X'20'               .. FLUSH VSE/POWER JOB
WDACCHG  EQU   X'10'               .. CHANGE STATEMENT
WDFLAG   DS    X'00'               FLAG BYTE
WDFLINS  EQU   X'80'               .. INSERT DONE
WDFLPOW  EQU   X'40'               .. FLUSH VSE/POWER JOB TO DO
WDAREA   DS    CL40' '             WORK AREA
```

```
WDLEN    EQU   *-WDSECT            LENGTH OF WORK AREA                00029000
         SPACE 2                                                      00029100
         IPW$EQU ,                 DEFINE REGISTER EQUATES AS         00029200
*                                  USED IN VSE/POWER CODING           00029300
         SPACE 2                                                      00029400
         IPW$DPA ,                 LAYOUT OF COMMON ADDRESS TABLE     00029500
*                                  (CAT) ALSO CALLED PERMANENT AREA   00029600
*                                  OF VSE/POWER.                      00029700
         SPACE 2                                                      00029800
         IPW$DQC ,                 LAYOUT OF DISK MANAGEMENT BLOCK    00029900
         SPACE 2                                                      00030000
         IPW$DQR ,                 LAYOUT OF INTERNAL QUEUE RECORD    00030100
         SPACE 2                   ADDRESSED BY TCQV OF TCB           00030200
         IPW$DTC ,                 LAYOUT OF TASK CONTROL BLOCK       00030300
         SPACE 2                                                      00030400
*        END                       NOT REQUIRED FOR '.A' COPY BOOK
```

Exit Routine for Output (Type OUTEXIT)

Function

To customize VSE/POWER's output processing to a greater extent than it is possible by VSE/POWER itself, the program provides an exit for a user routine in which you can modify every output record before it is being printed for:

1. local printing or punching, or
2. transmitting to an RJE workstation, or
3. passing data records to a device driving system, or
4. passing data records to a Spool-Access task that GETs output from the list or punch queue (provided the support has been enabled by the SET OUTEXIT=SAS autostart statement).

The general purpose of a user routine for output is:

- to modify data records to national standards needs
- to achieve different types of printer operation
- to produce own separator pages/cards
- to place the company's logo on a header page
- to produce accounting information on a trailer page
- to selectively produce separator pages for a particular user or job class
- to append security classification on every page.

The User Routine Work Area

To help you in making this output routine reentrant, VSE/POWER passes a work area to the routine to hold variables and information needed with the queue entry or to hold records to be inserted.

As described below in “Interface Description” on page 328, you find the address of this work area (for dump or trace reading) in the field OEXWA of the output exit parameter list.

You define the work area size in the POWER macro or in the PLOAD command (maximum 65,535 bytes, default zero) along with the routine phase name. VSE/POWER reserves this storage at start print time and releases the storage when the queue entry is printed. That means the work area is only available during printing of a queue entry and at print start time it always contains X'00'. The first four bytes of the work area contain its length. This information is refreshed every time the user routine gets control.

Restrictions

The output user routine should not invoke z/VSE macros which may cause, either voluntarily or involuntarily, a wait for an event or a resource. In particular, do not use a WAIT macro. When writing console messages, refer to the usage of the VSE/POWER IPW$WTO macro in "JOBEXIT Programming Example" on page 322.

Note: The exit routine must **not** use access-register mode and all executable code must be located below the 16MB line. The exit routine is executed in 24-bit addressing mode and must call VSE/POWER services and return to VSE/POWER in 24-bit addressing mode.

When the exit routine loses control (for example, due to a page fault), the status of the currently used addressing mode and the access registers are not saved. Thus, when the exit routine gets control back again, the previously used access registers cannot be restored.

Because the exit routine must not use the access-register mode, the exit routine cannot use data spaces. VSE/POWER does not accept an exit routine which has already been loaded into the SVA.

Interface Description

Use of Registers

This is the register content at entry to the output user routine:

```
Register 1:  address of parameter list described below
Register 14: return address
Register 15: entry point address of user routine
```

The user routine has access to all control blocks available in the environment; the pointers to these control blocks are passed as follows:

```
Register 5:  Queue record
Register 11: TCB
```

However, the user routine must not alter any fields in any control block to which it has access! Furthermore, the layout of these control blocks is subject to change in any future release of VSE/POWER.

The routine may not alter the content of registers 10, 11 and 12. These registers are reserved for VSE/POWER use only! All other registers may be used by the user routine.

The Parameter List

The list contains information about the passed records. A DSECT is provided by IPW$DXE, a macro without operands.

Table 81. Output Exit Parameter List

Name	Length	Type	Content
OEXRV	4	B	Record address
OEXRL	4	B	Record length
OEXCC	1	B	Operation Code (used as carriage control character)
OEXRT	1	B	Record Type: X'80' Job Header Record (1) X'40' Job Trailer Record (1) X'20' Data Set Header Record (1) X'08' Record of start separator page(1) X'04' Record of end separator page (1) X'00' Data or control record (1)
OEXTT	1	B	Task type: X'80' Local list task X'40' Local punch task X'20' RJE task X'02' Device service task X'01' Spool-Access GET task
OEXOT	1	B	Additional information: X'80' List output X'40' Punch output X'20' Start of queue entry X'10' Start next copy X'08' Queue entry processed X'04' Psetup command active
OEXWA	4	B	Address of work area
OEXRC	1	B	Return code from exit: X'00' Process record X'04' Delete record X'08' Insert record X'10' Flush queue entry X'18' Flush queue entry with hold X'1C' Stop task

Note: (1) Depends on the task type, whether certain records are passed (+) to the exit or not (-):

Table 82. Record Types Passed to an OUTEXIT

	Task Type				
Record Type	LST/PUN	RJE	Device Service (DST)	Spool Access Support (SAS)	Spool Access Support (SAS) (SET OUTEXIT=SAS)
Job Header Record	-	-	+	-	+
Data Set Header Record (*)	+	+	+	-	+
Job Trailer Record	(+)	(+)	+	-	+
(All other records or control records)	+	+	+	-	+

(*) - please refer to the *VSE/POWER Networking*, SC34-2603 and note the section "Sample of a PNET Receiver Exit Routine" when you have to process the various sections of a data set header record, especially the Output Processing Section, which contains 'user defined keywords' in an internally encoded form — called OPTB (also OPTU).

(+) - see “Processing of Queue Entries” on page 331 for details.

Fields set by VSE/POWER:

```
OEXRV  record address
OEXRL  record length
OEXCC  op code or carriage control character
OEXRT  record type
OEXTT  task type
OEXOT  various info
OEXWA  work area address
```

Field set by the user for normal processing, deletion of records, flush and stop processing:

```
OEXRC  return code from exit routine
```

Fields set by the user for insertion of records:

```
OEXRV  record address
OEXRL  record length
OEXCC  op code or carriage control
OEXRT  record type
OEXRC  return code from exit routine
```

You must set only the record types (OEXRT) X'08' or X'04' for separator pages. Any other record type you insert is handled by VSE/POWER as normal data or control records with the type code X'00'.

Make sure that OEXRT does not conflict with the return code set in OEXRC.

For OEXCC, do not use FF, FE, or FD as carriage control characters or your routine will terminate.

Return Codes

The exit routine must return control to VSE/POWER with one of the following return codes set in the parameter list field OEXRC.

X'00' Process record passed to the output exit; the exit routine may update fields within the boundaries of the record but may not change its length or address. If you change fields in any VSE/POWER control record (job header record or data set header record), this will have no effect! If you change data, check that you will not exceed the record length indicated in field OEXRL. Only changing of FCB and UCB name in the data set header record or changing the SETPRT parameter list record takes effect, because these values are interpreted after return from the output exit. Message 1R32I with RC=0003 is issued if record address or record length is changed.

X'04' Delete this record. Job header, job trailer, data set header record and end of data indication ***cannot*** be deleted. Message 1R32I with RC=0004 is issued to indicate this failure. The SETPRT parameter list record may be deleted.

X'08' Insert and process a new record. VSE/POWER returns the original record to the user routine once more. Any number of output records can be inserted.

The address, length, and operation code must be provided in the appropriate fields of the parameter list. A record with record length or record address of X'00' is rejected. A record length greater than 512 bytes is only allowed for CPDS records. VSE/POWER truncates all other records to 512 bytes without warning. Make sure that the record length you pass in OEXRL can be handled by the output device or device driving system. A CPDS record is only allowed for a device service task or Spool-Access GET task.

Job header, job trailer and data set header record, end of page and SETPRT parameter list records ***cannot*** be inserted. Message 1R32I with RC=0005 is issued if the user exit routine inserts an illegal record.

Note: VSE/POWER accepts every operation code (no check will be done) but only machine control characters for output make sense here since ASA conversion takes place earlier. Make sure that you insert valid operation codes for the appropriate type of output.

Note: When supporting a Device Service Task or a Spool Access Support Task it is not recommended to insert user records before the Job Header Record or the Data Set Header Record - otherwise the SAS application program may be confused when receiving an extra Data Set Reader record as an Inline SPL.

X'10' Flush output queue entry. The affected queue entry is retained in the output queue with disposition 'L' if the original disposition was 'K' or is deleted if the original disposition was 'D'. For a device service task a PFLUSH DEV,... is issued. Ignored for RJE and Spool-Access GET tasks.

X'18' Flush and Hold queue entry. The affected queue entry is retained in the output queue with disposition 'L' if the original disposition was 'K' or with 'H' if the original disposition was 'D'. For a device service task, a PFLUSH DEV,...,HOLD is issued. Ignored for RJE and Spool-Access GET tasks.

X'1C' Stop writer task. The affected queue entry is retained in the output queue with its original disposition.

Any other return code received from the output routine leads to a stop of the task and to requeue the queue entry with its original disposition. Message 1R32I with RC=0001 reflects this handling.

Processing of Queue Entries

Every queue entry which can be processed by a user written exit routine consists of a job header record, followed by a data set header record, followed by data records, followed by a job trailer record. Job header record and data set header record contain information about the job itself and about the data of the job. The job trailer record marks the end of the queue entry.

The exit routine can expect a job header record if it is working for a device service task or Spool-Access GET task. For the other task types no job header record is passed to the output exit. The 'start of queue entry'(X'20' in OEXOT) indication will therefore be set in the parameter list together with the very first record of the queue entry, independent of job header record or normal data record.

Similarly the job trailer record can also only be expected if the exit routine is working for a device service task or Spool-Access GET task. For the other task types, VSE/POWER passes a NOP record (X'03' in OEXCC) instead. Therefore, for all tasks, an additional artificial end of entry indication is passed to the output exit by a record address of X'00' and the 'queue entry processed' (X'08' in OEXOT) indication is set in the accompanying parameter list.

If a local list task is working for a 3800 printer, VSE/POWER does not pass a data set header record to the exit routine. Instead, VSE/POWER builds the 3800 SETPRT parameter list record (from the data set header record) and passes this record to the output exit.

If the PSETUP command is active for a list task, the output exit is informed via the X'04' indication in OEXOT. When PSETUP has finished, also 'queue entry processed' is indicated in the parameter list and processing of the queue entry is resumed at the beginning of the entry.

If multiple copies are produced, VSE/POWER indicates the start of every copy via a bit in the parameter list. When all copies are processed, VSE/POWER also passes 'queue entry processed' together with the record address of X'00'.

Accounting for the Output Exit Routine

The following account record types contain each a field showing the number of lines inserted and a field containing the number of records deleted by the routine:

```
the list account record
the punch account record
the spool operation account record
```

Device Driving System Considerations

A device driving system requests records from VSE/POWER via the Spool-Access Support GET interface handled by a device service task. Every record passed to the device driving system is preceded by a prefix containing the record number. Some products connected to VSE/POWER as a device driving system are sensitive to this record number (PSF for example). Deletion of records by the user written exit and the resulting gap in the record numbers causes no problems. But insertion of records leads to unpredictable results in the print pages prepared by the product in case of an error because inserted records are not written on spool. Therefore, insertion of records by the output exit called for a device service task must be handled carefully.

The flush return code issued by the output exit leads to queue a PFLUSH order control record (with or without the HOLD option) for the device driving system, that means, processing continues with the same queue entry until the device driving system reacts to the flush device order.

A stop request issued by the exit leads to set stop code 'S' for the device service task. This means that a return/feedback code (10/05) is given to the device driving system and the connection is terminated via a disconnect purge request.

Restart Considerations

Since the output exit can insert or delete records (which are not recorded on the VSE/POWER spool files), RESTART issued by the operator could lead to unexpected results. Example: A PDISPLAY command for a particular entry in the LST queue shows a page count of 4. Since records might have been inserted by the user exit routine, it is possible that currently page 7 is on the printer. If the operator wants to restart on page 6, message 1Q42I is displayed and the restart request is ignored.

Tracing of Exit Failures

When debugging logic failures of your exit routine, it may be helpful to obtain a snapshot dump of the VSE/POWER partition at a predefined processing point of your routine. You should include the call of macro IPW$IDM into the exit routine to have an Idump generated in flight. For details, refer to *VSE/POWER Networking*, SC34-2603.

OUTEXIT Programming Example

The sample of the OUTEXIT exit routine shown here is delivered to you as an A-book in PRD1.MACLIB under the name of OUTEXAMP.A. If you extend this example further, you may need macros that are only available with the optional code material.

The sample includes:

1. The set of statements that causes the source code of the OUTEXIT user exit routine to be assembled and cataloged.
2. The source code of the routine example itself.

Control Statements to Assemble and Catalog the Routine

```
* $$ JOB JNM=OUTEXRUN,CLASS=A,DISP=D
// JOB OUTEXRUN
// OPTION CATAL
// LIBDEF *,SEARCH=PRD1.MACLIB
// LIBDEF *,CATALOG=...
*
* PROVIDE FOR ... CATALOG SUBLIBRARY FOR OUTEXAMP
*
// EXEC ASSEMBLY,SIZE=100K
         COPY OUTEXAMP
         END
/*
// EXEC LNKEDT
/&
* $$ EOJ
```

Programming Example Source Code

```
         TITLE ' OUTEXAMP'                                              00001000
         PUNCH ' PHASE OUTEXAMP,*'                                      00002000
*********************************************************************** 00003000
**                                                                   ** 00004000
**                     O U T E X A M P                               ** 00005000
**                                                                   ** 00006000
**          VSE/POWER OUTPUT EXIT:  EXAMPLE PROGRAM        DY45811  ** 00007000
**                                                                   ** 00008000
**                                                                   ** 00009000
*********************************************************************** 00010000
*                                                                     * 00011000
*   THIS PROGRAM - NAMED OUTEXAMP - ACTS AS AN EXAMPLE FOR A USER     * 00012000
*   WRITTEN OUTPUT EXIT ROUTINE.                                      * 00013000
*                                                                     * 00014000
```

```
*   THIS EXAMPLE GIVES A SKELETON WHICH SHOWS HOW THE INTERFACE       *
*   BETWEEN VSE/POWER AND A USER WRITTEN OUTPUT EXIT WORKS.           *
*   IT CAN NOT SHOW THE VARIOUS FUNCTIONS AN OUTPUT EXIT COULD        *
*   PERFORM SINCE THAT DEPENDS ON THE NEEDS OF THE CUSTOMER.          *
*                                                                     *
*   THE FUNCTIONS, FOR EXAMPLE, COULD BE:                             *
*                                                                     *
*        - MODIFY DATA RECORDS TO NATIONAL STANDARDS' NEEDS           *
*        - PRODUCE OWN SEPARATOR PAGES/CARDS                          *
*        - PLACE THE COMPANY'S LOGO ON A HEADER PAGE                  *
*        - ACHIEVE DIFFERENT PRINTER OPERATIONS                       *
*        - APPEND SECURITY CLASSIFICATION ON EACH PAGE                *
*        - SELECTIVELY PRODUCE SEPARATOR PAGES/CARDS FOR A            *
*          PARTICULAR USER OR JOB                                     *
*                                                                     *
*   GIVEN HERE IS AN EXAMPLE HOW THE FUNCTION                         *
*   'ADD SECURITY CLASSIFICATION  ON EACH PAGE' CAN BE                *
*   IMPLEMENTED. HERE THE SECURITY CLASSIFICATION IS                  *
*   PRINTED ON TOP OF EVERY PAGE BY A LOCAL LIST TASK.                *
*                                                                     *
*   SINCE MORE THAN ONE LIST TASK CAN BE ACTIVE IN PARALLEL           *
*   A WORK AREA OF 50 BYTES IS REQUESTED BY THIS EXAMPLE.             *
*   THE WORK AREA IS USED HERE TO SAVE RECORD INFORMATION.            *
*   IF THE WORK AREA IS NOT PRESENT THE LIST TASK IS STOPPED.         *
*                                                                     *
*                                                                     *
*                                                                     *
*   THE FOLLOWING ADDRESSABILITY IS ASSUMED AT ENTRY TO THE           *
*   OUTPUT EXIT:                                                      *
*                                                                     *
*        R1 - ADDRESS OF PARAMETER LIST PASSED BY VSE/POWER           *
*        RA - ADDRESS OF VSE/POWER COMMON ADDRESS TABLE (CAT)         *
*        RB - ADDRESS OF TASK CONTROL BLOCK (TCB)                     *
*        RE - RETURN ADDRESS TO VSE/POWER                             *
*        RF - BASE REGISTER OF THIS ROUTINE                           *
*                                                                     *
*   THE OUTPUT EXIT MAY NOT ALTER THE CONTENTS OF                     *
*   REGISTERS 10, 11 AND 12. THESE REGISTERS ARE RESERVED FOR         *
*   VSE/POWER. ALL OTHER REGISTERS MAY BE USED BY                     *
*   THE OUTPUT EXIT.                                                  *
*                                                                     *
*   THIS OUTPUT EXIT PHASE IS LOCATED WITHIN THE                      *
*     - PAGEABLE AREA OF THE VSE/POWER PARTITION WHEN LOADED          *
*       AT VSE/POWER INITIALIZATION TIME, OR IN THE                   *
*     - GETVIS AREA OF THE VSE/POWER PARTITION WHEN LOADED            *
*       AFTER INITIALIZATION BY THE 'PLOAD' COMMAND.                  *
*   FOR DEBUGGING LOCATE THE OUTPUT EXIT IN STORAGE BY                *
*   THE STORAGE DESCRIPTOR 'OUTPUT-EXIT' AND BY THE POINTER 'CAOEX'   *
*   IN THE COMMON ADDRESS TABLE OF VSE/POWER.                         *
*                                                                     *
***********************************************************************
*                                                                     *
***********************************************************************
*                                                                     *
*   THE FOLLOWING MACROS ARE REQUIRED:                                *
*                                                                     *
*        VSE/POWER:                                                   *
*                                                                     *
*           IPW$DXE - DSECT FOR THE INTERFACE PARAMETER LIST          *
*                                                                     *
*                                                                     *
*                                                                     *
***********************************************************************
         EJECT
         SPACE 2
*        REGISTER USAGE
         SPACE 2
```

```
*        R0 - **** - WORK REGISTER                                      00082000
*        R1 - **** - POINTER TO INTERFACE PARAMETER LIST                00083000
*        R2 - **** - WORK REGISTER                                      00084000
*        R3 - **** - WORK REGISTER                                      00085000
*        R4 - **** - WORK REGISTER                                      00086000
*        R5 - **** - POINTER TO QUEUE RECORD (NOT USED BY OUTPUT EXIT)  00087000
*        R6 - **** - POINTER TO THE WORK AREA                           00088000
*        R7 - **** - WORK REGISTER                                      00089000
*        R8 - **** - WORK REGISTER                                      00090000
*        R9 - **** - WORK REGISTER                                      00091000
*        RA - **** - POINTER TO CAT (NOT USED BY OUTPUT EXIT)           00092000
*        RB - **** - POINTER TO TCB                                     00093490
*        RD - **** - WORK REGISTER                                      00094000
*        RE - **** - RETURN ADDRESS TO VSE/POWER                        00095000
*        RF - **** - BASE REGISTER                                      00096000
         EJECT                                                          00097000
OUTEXAMP DS    0H                                                       00098000
         USING *,RF                ESTABLISH ADDRESSABILITY             00099000
         USING PADS,RA             MAKE VSE/POWER CAT ADDRESSABLE       00100000
         USING TCDS,RB             MAKE VSE/POWER TCB ADDRESSABLE       00101000
         USING OEXDS,R1            ESTABLISH ADDRESSABILITY FOR         00102000
*                                  PARAMETER LIST                       00103000
         B     IPWX005             BRANCH OVER STORAGE DESCRIPTOR       00104000
         SPACE                                                          00105000
         DC    CL12'OUTPUT-EXIT'   DEFINE STORAGE DESCRIPTOR            00106000
         SPACE                                                          00107000
*              SINCE THIS OUTPUT EXIT EXAMPLE NEEDS A WORK AREA         00108000
*              THE TASK WILL BE STOPPED IF NONE IS PRESENT.             00109000
         SPACE 2                                                        00109100
*        WHENEVER A WORK AREA IS USED, THE LENGTH OF THIS AREA MUST BE  00109200
*        VERIFIED EACH TIME THE EXIT GAINS CONTROL, BECAUSE             00109300
*        - THE EXIT MIGHT HAVE BEEN LOADED BY OPERATOR WITH A WRONG     00109400
*          LENGTH OF ITS WORK AREA                                      00109500
*        - A NEW VERSION OF THE EXIT MIGHT HAVE BEEN LOADED AND THE     00109600
*          PREVIOUSLY DEFINED WORK AREA IS STILL USED UNTIL THE TASK    00109700
*          ENDS.                                                        00109800
         SPACE                                                          00110000
IPWX005  DS    0H                                                       00111000
         ICM   R6,15,OEXWA         WORK AREA PRESENT ?                  00112000
         BZ    IPWX007             ..NO, STOP TASK                      00113490
         USING WDSECT,R6           ESTABLISH ADDRESSABILITY FOR WORK    00114000
*                                  AREA                                 00115000
*        THE WORK AREA EXISTS AND CAN NOW BE ADDRESSED. TEST IF         00115050
*        THE WORK AREA IS LARGE ENOUGH.                                 00115100
         SPACE 1                                                        00115150
         L     R7,WDLENGTH             GET SPECIFIED WORK AREA SIZE     00115200
         LA    R8,WDLEN                GET REQU. SIZE OF WORK AREA      00115250
         CR    R7,R8                   WORK AREA TOO SHORT ?            00115300
         BNL   IPWX010                 ..NO, CONTINUE EXIT              00115350
         SPACE 1                                                        00115400
IPWX007  DS    0H                                                       00115450
         SPACE 1                                                        00115500
*        RESPECT THAT MACRO IPW$WTO DESTROYS REGISTER R0-R3             00115550
         SPACE 1                                                        00115600
         LA    R7,IPWMSG1          GET ADDRESS OF IPWMSG1               00115650
         BAL   R6,SUBWTO           ISSUE OEXMSG1                        00115700
         SPACE 1                                                        00115750
         LA    R7,IPWMSG2          GET ADDRESS OF IPWMSG2               00115800
         BAL   R6,SUBWTO           ISSUE OEXMSG2                        00115850
         SPACE 2                                                        00115900
         B     IPWXSTP             GO AND STOP TASK                     00115950
         SPACE                                                          00116000
*              DIFFERENT ACTIONS MAY BE NECESSARY FOR THE DIFFERENT     00117000
*              TYPES OF TASK. THEREFORE WE DETERMINE THE TYPE OF TASK   00118000
*              FOR WHICH THE OUTPUT EXIT CURRENTLY WORKS.               00119000
         SPACE                                                          00120000
IPWX010  DS    0H                                                       00121000
```

```
         TM    OEXTT,OETLST        WORKING FOR A LOCAL LIST TASK ?
         BO    IPWX100             ..YES,CONTINUE
         SPACE
         TM    OEXTT,OETPUN        WORKING FOR A LOCAL PUNCH TASK ?
         BO    IPWX200             ..YES, CONTINUE
         SPACE
         TM    OEXTT,OETDST        WORKING FOR DEVICE SERVICE TASK ?
         BO    IPWX300             ..YES, CONTINUE
         SPACE
         TM    OEXTT,OETXPT        WORKING FOR SPOOL ACCESS GET TASK ?
*                                  ..IN CASE 'SET OUTEXIT=SAS'
         BO    IPWX350             ..YES, CONTINUE
         SPACE
         B     IPWX400             MUST BE RJE TASK
         SPACE
***********************************************************************
**                HANDLE LOCAL LIST TASK                             **
***********************************************************************
*                                                                     *
*   FOR A LOCAL LIST TASK A SECURITY CLASSIFICATION IS PRINTED        *
*   AS FIRST RECORD ON EACH PAGE. A NEW PAGE ALWAYS STARTS AFTER      *
*   POSITIONING THE PRINTER VIA A 'SKIP TO CHANNEL 1' COMMAND         *
*   (OPERATION CODE X'8B') TO THE FIRST LINE OF THE PAGE.             *
*   THAT MEANS AFTER THE 'SKIP TO CHANNEL 1' WAS PASSED TO THE        *
*   PRINTER A NEW RECORD (THE RECORD WITH THE SECURITY                *
*   CLASSIFICATION) MUST BE INSERTED.                                 *
*                                                                     *
***********************************************************************
         SPACE 2
IPWX100  DS    0H
         CLI   OEXCC,IPW8B         SKIP TO CHANNEL 1 RECEIVED ?
         BNE   IPWX110             ..NO, CONTINUE
         SPACE
*              THE CURRENT OPERATION CODE IS SAVED IN THE WORK AREA TO
*              KNOW THAT SKIP TO CHANNEL 1 IS PROCESSED.
         SPACE
         MVC   WDOPCODE,OEXCC      SAVE CURRENT OPERATION CODE
         B     IPWXNOR             CONTINUE NORMAL PROCESSING
         SPACE
IPWX110  DS    0H
         CLI   WDOPCODE,IPW8B      WAS LAST RECORD SKIP TO CHANNEL 1 ?
         BNE   IPWXNOR             ..NO, CONTINUE NORMAL PROCESSING
         SPACE
*              SET UP PARAMETER LIST TO INSERT RECORD
         SPACE
         LA    R2,IPDATA1          GET RECORD ADDRESS
         ST    R2,OEXRV            SET UP RECORD ADDRESS
         LA    R2,L'IPDATA1        GET LENGTH OF RECORD
         ST    R2,OEXRL            SET UP RECORD LENGTH
         MVI   OEXCC,IPW11         SET UP OPERATION CODE
         MVI   WDOPCODE,X'00'      CLEAR OP CODE TO FORGET ABOUT
*                                  SKIP TO CHANNEL 1
         B     IPWXINS             CONTINUE WITH INSERT
         SPACE 2
***********************************************************************
**                HANDLE LOCAL PUN TASK                              **
***********************************************************************
         SPACE 2
IPWX200  DS    0H
*                 INSERT HERE SPECIAL FUNCTIONS TO BE PERFORMED FOR
*                 A PUNCH TASK
         SPACE
         B     IPWXNOR             CONTINUE NORMAL PROCESSING
         SPACE 2
***********************************************************************
**                HANDLE DEVICE SERVICE TASK                         **
***********************************************************************
```

```
         SPACE 2                                                        00189000
IPWX300  DS    0H                                                       00190000
*                INSERT HERE SPECIAL FUNCTIONS TO BE PERFORMED FOR      00191000
*                A DEVICE SERVICE TASK                                  00192000
         SPACE                                                          00193000
         B     IPWXNOR              CONTINUE NORMAL PROCESSING          00194000
         SPACE 2                                                        00195000
*********************************************************************** 00196000
**               HANDLE SPOOL ACCESS GET TASK                        ** 00197000
*********************************************************************** 00198000
         SPACE 2                                                        00199000
IPWX350  DS    0H                                                       00200000
*                INSERT HERE SPECIAL FUNCTIONS TO BE PERFORMED FOR      00201000
*                A SPOOL ACCESS GET TASK                                00202000
         SPACE                                                          00203000
         B     IPWXNOR              CONTINUE NORMAL PROCESSING          00204000
         SPACE 2                                                        00205000
*********************************************************************** 00206000
**               HANDLE RJE TASK                                     ** 00207000
*********************************************************************** 00208000
         SPACE 2                                                        00209000
IPWX400  DS    0H                                                       00210000
*                INSERT HERE SPECIAL FUNCTIONS TO BE PERFORMED FOR      00211000
*                AN RJE TASK                                            00212000
         SPACE                                                          00213000
         B     IPWXNOR              CONTINUE NORMAL PROCESSING          00214000
         SPACE 2                                                        00215000
*********************************************************************** 00216000
**               STOP TASK                                           ** 00217000
*********************************************************************** 00218000
         SPACE 2                                                        00219000
IPWXSTP  DS    0H                                                       00220000
         MVI   OEXRC,OERSTP        INDICATE TO STOP THE TASK            00221000
         B     IPWXEXT             CONTINUE                             00222000
         SPACE 2                                                        00223000
*********************************************************************** 00224000
**               FLUSH HOLD QUEUE ENTRY (NOT USED BY THIS EXAMPLE)  ** 00225000
*********************************************************************** 00226000
         SPACE 2                                                        00227000
IPWXFLH  DS    0H                                                       00228000
         MVI   OEXRC,OERFLH        INDICATE TO FLUSH HOLD THE QUEUE     00229000
*                                  ENTRY                                00230000
         B     IPWXEXT             CONTINUE                             00231000
         SPACE 2                                                        00232000
*********************************************************************** 00233000
**               FLUSH QUEUE ENTRY  (NOT USED BY THIS EXAMPLE)       ** 00234000
*********************************************************************** 00235000
         SPACE 2                                                        00236000
IPWXFLS  DS    0H                                                       00237000
         MVI   OEXRC,OERFLS        INDICATE TO FLUSH THE QUEUE ENTRY    00238000
         B     IPWXEXT             CONTINUE                             00239000
         SPACE 2                                                        00240000
*********************************************************************** 00241000
**               INSERT A NEW RECORD                                 ** 00242000
*********************************************************************** 00243000
         SPACE 2                                                        00244000
IPWXINS  DS    0H                                                       00245000
         MVI   OEXRC,OERINS        INDICATE TO INSERT A NEW RECORD      00246000
         B     IPWXEXT             CONTINUE                             00247000
         SPACE 2                                                        00248000
*********************************************************************** 00249000
**               DELETE THE CURRENT RECORD (NOT USED BY THIS         ** 00250000
**                                           EXAMPLE)                ** 00251000
*********************************************************************** 00252000
         SPACE 2                                                        00253000
IPWXDEL  DS    0H                                                       00254000
         MVI   OEXRC,OERDEL        INDICATE TO DELETE THE RECORD        00255000
```

```
         B     IPWXEXT             CONTINUE                              00256000
         SPACE 2                                                         00257000
*********************************************************************** 00258000
**               NORMAL RETURN                                       ** 00259000
*********************************************************************** 00260000
         SPACE 2                                                         00261000
IPWXNOR  DS    0H                                                        00262000
         MVI   OEXRC,OEROK         INDICATE NORMAL PROCESSING            00263000
         SPACE 2                                                         00264000
*********************************************************************** 00265000
**               EXIT                                                ** 00266000
*********************************************************************** 00267000
         SPACE 2                                                         00268000
IPWXEXT  DS    0H                                                        00269000
         BR    RE                  RETURN TO VSE/POWER                   00270000
         SPACE                                                           00271000
         DROP  R1,R6               RELEASE ADDRESSABILITY                00272000
         SPACE                                                           00273000
         EJECT                                                           00274000
*********************************************************************** 00274030
*        SUBROUTINE TO ISSUE MESSAGE ON OPERATOR CONSOLE              * 00274060
*                                                                     * 00274090
* USAGE OF VSE/POWER MACRO IPW$WTO:                                   * 00274120
*       THIS MACRO, TOGETHER WITH SOME INDICATIONS SET IN THE TCB,    * 00274150
*       CAN BE USED TO DISPLAY A MESSAGE ON THE CENTRAL OPERATOR      * 00274180
*       CONSOLE.                                                      * 00274210
*       THE USAGE OF THIS MACRO DESTROYS THE REGISTER R0 - R3.        * 00274240
*       THE MESSAGE TO BE DISPLAYED SHOULD CONTAIN ALPHAMERIC         * 00274270
*       CHARACTERS ONLY.                                              * 00274300
*                                                                     * 00274330
*********************************************************************** 00274360
         SPACE 2                                                         00274390
SUBWTO   DS    0H                                                        00274420
         STCM  R7,7,TCMW+1             PASS MESSAGE ADDRESS              00274450
         MVI   TCMW,X'00'              CLEAR OPTION BYTE                 00274480
         OI    TCMW,TCPCOP             FORCE MESSAGE TO CONSOLE          00274510
         OI    TCMW,TCDNMM             SUPPRESS MESSAGE MODIFICATION     00274540
         SPACE 1                                                         00274570
         IPW$WTO TCMW                  ISSUE MESSAGE                     00274600
         SPACE                                                           00274630
         MVI   TCMW,X'00'              CLEAR OPTION BYTE                 00274660
         BR    R6                      RETURN TO CALLER                  00274690
         SPACE 5                                                         00274720
         DROP  RA,RB               RELEASE ADDRESSABILITY                00274750
         EJECT                                                           00274780
         SPACE                                                           00274810
*********************************************************************** 00275000
**               D E F I N I T I O N S                               ** 00276000
*********************************************************************** 00277000
         SPACE 2                                                         00278000
IPW8B    EQU   X'8B'                OP. CODE: SKIP TO CHANNEL 1          00279000
IPW11    EQU   X'19'                OP. CODE: WRITE AND SPACE 3 LINES    00280000
         SPACE                                                           00281000
IPWMSG1  DC    AL1(IPWMSG1L-IPWMSG1-1)   LENGTH OF MESSAGE               00281100
         DC    C'INCORRECT SIZE OF WORKAREA GIVEN, 50 BYTES NEEDED'      00281200
IPWMSG1L EQU   *                                                         00281300
         SPACE 1                                                         00281400
IPWMSG2  DC    AL1(IPWMSG2L-IPWMSG2-1)   LENGTH OF MESSAGE               00281500
         DC    C'DISABLE OUTEXIT USING PVARY'                            00281600
IPWMSG2L EQU   *                                                         00281700
         SPACE                                                           00281800
*********************************************************************** 00282000
*        LINE TO BE INSERTED BY THE OUTPUT EXIT                       * 00283000
*********************************************************************** 00284000
         SPACE                                                           00285000
IPDATA1  DC    C'          ******* INTERNAL USE ONLY *******          ' 00286000
         SPACE                                                           00287000
```

```
*********************************************************************** 00288000
*        DSECT FOR WORK AREA                                          * 00289000
*********************************************************************** 00290000
         SPACE                                                          00291000
WDSECT   DSECT                      DSECT OF WORK AREA                  00292000
WDLENGTH DS    F                    LENGTH OF WORK AREA                 00293000
WDOPCODE DS    X'00'                SAVE OPERATION CODE                 00294000
WDAREA   DS    CL45' '              UNUSED PART OF WORK AREA            00295000
WDLEN    EQU   *-WDLENGTH           LENGTH OF WORK AREA                 00295500
         SPACE                                                          00296000
         EJECT                                                          00297000
*********************************************************************** 00298000
         LTORG                                                          00299000
*********************************************************************** 00300000
*********************************************************************** 00301000
*        DUMMY SECTION OF  PARAMETER LIST                             * 00302000
*********************************************************************** 00303000
         SPACE                                                          00304000
         IPW$DXE                                                        00305000
         SPACE 4                                                        00306000
         IPW$EQU ,                  DEFINE REGISTER EQUATES AS          00307000
*                                   USED IN VSE/POWER CODING            00308000
         SPACE 2                                                        00309000
         IPW$DPA ,                  LAYOUT OF COMMON ADDRESS TABLE      00310000
*                                   (CAT) ALSO CALLED PERMANENT AREA    00311000
*                                   OF VSE/POWER.                       00312000
         SPACE 2                                                        00313000
         IPW$DTC ,                  LAYOUT OF TASK CONTROL BLOCK (TCB)  00314000
         SPACE 2                                                        00315000
         IPW$DQR ,                  LAYOUT OF INTERNAL QUEUE RECORD     00316000
         SPACE 2                    ADDRESSED BY TCQV OF TCB            00317000
         IPW$DNR JHR=YES,JTR=YES,DHR=YES,OUT=YES NETWORK CONTROL RECS   00318000
         SPACE 2                                                        00319000
*        END                        NOT REQUIRED FOR '.A' COPY BOOK     00320000
```

Part 4. Appendixes

Appendix A. Cross-Partition Communication via Spool Macros

This chapter describes the XECB-macro based cross-partition communication (SPOOL macro) support.

Restriction

If VSE/POWER is running in a partition which is allocated in a private address space, the user program communicating to VSE/POWER must run in a partition which is allocated in the same address space as the VSE/POWER partition. For more information, see “Return Codes in Register 15” on page 359.

This support allows you to access VSE/POWER services from within a program. While it ensures program compatibility, you can use it side by side with the XPCC-macro based support described in Part 2, “Spool-Access Support,” on page 55. Continued use of the SPOOL macro support requires that, for VSE/POWER table generation, you specify SPOOL=YES in the POWER generation macro.

For using the SPOOL macro support, the macros described in this chapter are available.

In addition, you need the z/VSE macro XECBTAB, described in *z/VSE System Macros Reference*, SC34-2638.

To connect to VSE/POWER, use the z/VSE XECBTAB macro with the following operands:

```
►►──XECBTAB TYPE=DEFINE,XECB=──┬─SPMXECB─┬──,ACCESS=XWAIT──────────────────────►◄
                               └─ICRXECB─┘
```

Specify XECB=SPMXECB for a GETSPOOL or a CTLSPOOL macro, specify XECB=ICRXECB for a PUTSPOOL macro. An XECB (cross-partition event control block) must be at least eight bytes long.

VSE/POWER requires the three-byte address of a spool parameter list (SPL) to be inserted into the XECB before your program issues a service request. You insert this address at:

SPMXECB+5 for a GETSPOOL or a CTLSPOOL request

ICRXECB+5 for a PUTSPOOL request

Other than XECBTAB, no z/VSE macro is required for VSE/POWER's SPOOL macro support. Issue an XECBTAB=DELETE for the defined XECBs when the support is no longer required by your program.

Coding Practices

Only one user of the PUTSPOOL macro, and only one user of either the GETSPOOL or the CTLSPOOL macro may be active at any point in time. Therefore, in private multitasking environments, as for example in CICS, the user must provide an enqueuing mechanism to ensure that the spooling resource SPMXECB or ICRXECB is serialized. In addition, it is recommended to use the XECBTAB TYPE=DELETALL request in abend routines in order to have VSE/POWER informed. You can bypass this restriction and also avoid many a contention situation by using the support described in Chapter 6, "Introduction to Spool-Access Support," on page 57.

For the conventions used in presenting macro formats in this appendix, see Chapter 1, "Understanding Syntax Diagrams," on page 3. A coding example for using the SPOOL macro support is given under "Coding Example for Using the SPOOL Macros" on page 361.

Spool Access Protection Considerations

This mode of security protection can be activated when starting VSE/POWER if it was also enabled at IPL. This protection mode limits *eligible* spool entry access to *authenticated* users or programs, or to system administrators, i.e., when access is restricted to certain user IDs, these must be authenticated. Authentication requires a security logon with a password or a system component logon, such as IUI. This mode applies only when using GETSPOOL or CTLSPOOL.

A PXMIT command routed to a new local node will be tagged with the issuer's security user ID if Spool Access Protection is active, replacing the originator user ID identified in the SPL field SPUS.

If a PXMIT command is issued by a non-authenticated user, this is indicated in the command when it is routed to the target node. PXMIT commands from systems without the Spool Access Protection feature active (e.g., downlevel systems or non-VSE systems) will be assumed to be authenticated.

If VSE/POWER Spool Access Protection is active, then every attempt will be made to tag a job spool entry with an origin userid to obtain Spool Access Protection eligibility. In the case of a PUTSPOOL macro, the USERID=userid operand of the PUTSPOOL macro is optional and the origin userid may not available (field SPUS). In this case VSE/POWER will search for a security logon userid for the XECB PUTSPOOL program (available from the PUTSPOOL program's `* $$ JOB SEC=(userid,pwd)` or `// ID USER=userid,PWD=password` statement if any). If the security logon userid is available, then it will be used as the job origin userid.

Programs issuing the GETSPOOL/CTLSPOOL macro previously could access any spool entry without regard to the spool entry's matching origin or target userid(s). Now, if a spool entry has an origin or target userid, and the GETSPOOL/CTLSPOOL program does not have system administrator authority, the program must perform a security logon with the same origin or target userid to gain access, e.g. via `* $$ JOB SEC=(userid,pwd)` or via `// ID USER=cccccccc,PWD=password`. To obtain system administrator authority, the use of the VSE/POWER Master Password should be considered.

General Notes

1. VSE/POWER responds to spooling requests from the SVA. However, the required SPLs and data areas must reside in the partitions that contain the requesting programs.
2. A program using the SPOOL macro support must include an SPL TYPE=MAP macro.
3. The operand PBUF=buffaddr must be specified in either the definition macro SPL, or in the execution macro (CTLSPOOL, GETSPOOL, or PUTSPOOL) that is executed first in the program.
4. A system error may occur if:
   a. The partition using the SPOOL macro support has a higher priority than VSE/POWER.
   b. An abnormal end or shut down of VSE/POWER occurs before all active SPOOL macro service tasks have completed.
5. Before using the support, you must save your registers 0, 1, 13, 14, and 15 (they are used and overwritten by VSE/POWER). Register 15 has the return code.
6. The operands CLASS= and DISP= are not supported in the PUTSPOOL macro or its associated SPL.

   To specify the output class or disposition for a job submitted by PUTSPOOL, include an * $$ LST or * $$ PUN statement at the beginning of the job. If you supply an * $$ JOB statement, include the * $$ LST or * $$ PUN statement immediately behind the * $$ JOB statement.

   If these PUTSPOOL operands have already been coded in an existing program, VSE/POWER updates the SPL with the specified value (in case the SPL is used later by a GETSPOOL or CTLSPOOL macro).

   If you use these operands in a modified source program, then:
   - You receive a warning comment if they occur in the SPL macro.
   - You get an assembler generated MNOTE if they occur in the PUTSPOOL macro.
7. If you use the spooling macros in a private multitasking environment like CICS, you must provide your own queueing mechanism to ensure that the spooling resource is serialized.
8. If you use the 'STXIT AB' macro or the 'HANDLE ABEND' routines in CICS, you need to use the 'XECBTAB TYPE=DELETALL' macro.

SPL Macro: Generate a Spool Parameter List

The macro builds a spool parameter list (SPL) for use by the execution macros PUTSPOOL, GETSPOOL, and CTLSPOOL. Any specification you make in an SPL is in effect for the execution macro using this SPL, except if (a) the specification is overridden by a corresponding operand of the execution macro or (b) the REQ= operand is not specified in the CTLSPOOL macro; in that case the class specification is modified.

Correct use of the macro requires you to:
1. Store the SPL address into the correct XECB.
2. Load the pointer register named in the SPL=(reg) operand of the CTLSPOOL, GETSPOOL, or PUTSPOOL macro.

Formats of the Macro

The macro has the following two formats:

Format 1: Generating an SPL

```
                                                           ┌─,CLASS=A─────┐
►►──┬──────┬──SPL TYPE=DEFINE──┬────────────────────┬──┼──────────────┼──────────►
    └─name─┘                    └─,CBUF=firstbuffaddr─┘  └─,CLASS=class─┘

  ┌─,DISP=K───────────┐  ┌─,JOBN=DUMMY───┐
►─┼───────────────────┼──┼───────────────┼──┬───────────────┬────────────────────►
  └─,DISP=disposition─┘  └─,JOBN=jobname─┘  └─,NEWVAL=value─┘

                    ┌─,PBUFL=88──────────┐
►─┬───────────────┬─┼────────────────────┼─┬───────────────┬─────────────────────►
  └─,PBUF=buffaddr─┘ └─,PBUFL=bufflength─┘ └─,PWD=password─┘

►─┬──────────────────┬─┬─────────────────┬───────────────────────────────────────►◄
  └─,REQ=─┬─CANCEL──┬─┘ └─,USERID=user_id─┘
          ├─CLASS───┤
          ├─COMMAND─┤
          ├─DISP────┤
          ├─LOOKUP──┤
          ├─PRI─────┤
          ├─REMOTE──┤
          ├─SCRATCH─┤
          └─STATUS──┘
```

Format 2: Generating a DSECT

```
                               ┌─,ICRXECB=NO──┐  ┌─,SPMXECB=NO──┐
►►──┬──────┬──SPL TYPE=MAP──┼──────────────┼──┼──────────────┼──────────────►◄
    └─name─┘                 └─,ICRXECB=YES─┘  └─,SPMXECB=YES─┘
```

TYPE=DEFINE
: The operand causes an SPL to be set up with the specified values.

CBUF=firstbuffaddr
: The operand specifies the address of the first buffer of a chain of buffers that contain the job stream. Each buffer is 88 bytes long and has the following contents:

```
Bytes 0 - 3  = Pointer to the next buffer in the chain; set to
               zero in the last buffer
Bytes 4 - 7  = Reserved
Bytes 8 - 87 = Spool record
```

Up to 4095 such buffers may be chained for every PUTSPOOL access.

CLASS=A|class
: The operand specifies the VSE/POWER output class (A-Z) for the affected job. The operand is ignored if specified in an SPL for PUTSPOOL.

DISP=K|disposition
: The operand specifies the output disposition for the affected job. The operand is ignored if specified in an SPL for PUTSPOOL.

JOBN=DUMMY|jobname
The operand specifies the job name to be assigned to the affected input queue entry for a PUTSPOOL operation or to be searched for in case of a CTLSPOOL or GETSPOOL operation.

NEWVAL=value
The operand is meaningful only if the SPL is to be used with CTLSPOOL.

For value in the operand, specify the new value that is to be assigned in accordance with your specification in the REQ operand of the CTLSPOOL macro. You can specify a new value for one of the following:

```
Class of the job:        REQ=CLASS in CTLSPOOL
Disposition of the job:  REQ=DISP in CTLSPOOL
Priority of the job:     REQ=PRI in CTLSPOOL
A remote ID:             REQ=REMOTE in CTLSPOOL
```

The value can be specified in one of the following ways:

```
C'x'     For example, NEWVAL=C'A'
X'nn'    For example, NEWVAL=X'01'
```

PBUF=buffaddr
For buffaddr, specify the address of an area for use by VSE/POWER and for VSE/POWER feedback information under certain error conditions.

PBUFL=88|bufflength
For bufflength, specify (in number of bytes) the length of the buffer whose address is given in PBUF=buffaddr. Define your buffer's length large enough for your longest data record to fit into the buffer. VSE/POWER truncates the trailing blanks of a record; it indicates the length of every record after truncation in either:

- the four-byte SPL field SPRL if data records are not blocked, or
- bytes 2 and 3 of the record prefix if the data records are blocked (see also the MODE=BUF operand of the GETSPOOL macro).

The minimum length you can specify is 88.

PWD=password
The operand specifies the password to be associated with the request.

If the queue entry to be accessed by CTLSPOOL or GETSPOOL carries an explicit password (neither defaults of zero or blank), then specify this password.

If you want to create a password protected job, then supply this operand for the PUTSPOOL request.

If you omit this operand, a default blank password will be given to submitted jobs or will be used in combination with CTLSPOOL or GETSPOOL requests. This does not hinder the latter requests to gain access to locally read in jobs.

The password can be any alphameric string of up to eight characters.

REQ=CANCEL|CLASS|COMMAND|DISP|LOOKUP|PRI|REMOTE |SCRATCH|STATUS
The operand defines a default for CTLSPOOL requests. For the various specifications, refer to “Format of the Macro” on page 348.

USERID=user-id
For user-id specify an alphameric string of up to eight characters.

Supply such an owning user-id in the PUTSPOOL request when you want to prevent unauthorized CTLSPOOL access to the job or to its produced output.

If you want to manipulate an existing queue entry that shows an explicit from/to user-id then specify the corresponding user-id in your CTLSPOOL request.

For GETSPOOL requests, the user-id specification is not required.

TYPE=MAP
The operand causes a DSECT of the SPL to be generated. An SPL macro with TYPE=MAP must be specified at least once in a program using the SPOOL macro support.

ICRXECB=YES|NO
Specify ICRXECB=YES if the DSECT to be generated is to apply to an SPL for use with PUTSPOOL.

SPMXECB=YES|NO
Specify SPMXECB=YES if the DSECT to be generated is to apply to an SPL for use with CTLSPOOL or GETSPOOL.

CTLSPOOL Macro: Control VSE/POWER Jobs

The macro requests VSE/POWER to do one of the following:

- Alter the attributes of a VSE/POWER job.
- Cancel a submitted job prior to its execution.
- Delete the list or punch output of a job after its execution.
- Display the status of any job or of all jobs.
- Send a message to another user, remote operator, or central operator.
- Submit a VSE/POWER command for execution.

Nearly all of the macro's operands allow you to use register notation (indicated by "(reg)" as a possible specification). You can use for this purpose any register except 0, 1, 14, and 15.

Requirements for the Caller

AMODE:
24

RMODE:
24

ASC Mode:
Primary

Format of the Macro

```
>>-+------+--CTLSPOOL SPL=(reg)--+----------------------+--+---------------------+-->
   '-name-'                      '-,CCLASS=-+-value-+-'  '-,JNUM=-+-SPL---+-'
                                            '-(reg)-'             '-(reg)-'

>--+-----------------------+--+-------------+--+-----------------------+--------->
   '-,JOBN=-+-jobname-+-'    '-,MODE=SPOOL-'  '-,NEWVAL=-+-value-+-'
            '-(reg)---'                                  '-(reg)-'
```

```
                                                        ┌─,QUEUE=LST──────┐
►─┬────────────────────────┬─┬─────────────────────────┬─┼─────────────────┼──────────►
  └─,PBUF=─┬─buffaddr─┬────┘ └─,PWD=─┬─password─┬──────┘ └─,QUEUE=─┬─PUN─┬─┘
           └─(reg)────┘              └─(reg)────┘                  ├─RDR─┤
                                                                   └─XMT─┘

►─┬──────────────────────┬─┬───────────────────────────┬─────────────────────────────►◄
  └─,REQ=─┬─CANCEL──┬────┘ └─,USERID=─┬─user-id─┬──────┘
          ├─CLASS───┤                 └─(reg)───┘
          ├─COMMAND─┤
          ├─DISP────┤
          ├─LOOKUP──┤
          ├─PRI─────┤
          ├─REMOTE──┤
          ├─SCRATCH─┤
          ├─STATUS──┤
          └─(reg)───┘
```

SPL=(reg)
This mandatory operand specifies the register which contains the address of the spool parameter list (SPL) to be used. The SPL defines the request to VSE/POWER.

CCLASS=value|(reg)
For value, specify the class of the queue entries to which the CTLSPOOL request is to apply. You can specify the value in one of the following forms:

```
C'x'   For example, CCLASS=C'A'
X'nn'  For example, CCLASS=X'F1'
```

If you use a register, it must contain the class in its low-order byte. This operand is valid only with one of the following:

```
REQ=CLASS
REQ=DISP
REQ=PRI
REQ=REMOTE
```

JNUM=SPL|(reg)
The operand specifies the job number that is to be used as a search argument together with the job name.

Specify JNUM=SPL if VSE/POWER is to use the job number currently stored in the SPL.

If you use a register it must contain the job number.

If you omit the operand, VSE/POWER takes the first job with a matching name.

JOBN=jobname|(reg)
For jobname, specify the name by which the job is known to VSE/POWER.

If you use a register, it must contain a pointer to an eight-byte storage field containing the job's name.

MODE=SPOOL
The operand causes VSE/POWER to write its response to the CTLSPOOL request into the LST queue and to return the LST queue entry's job name and number in the SPL used for the request. Issue a GETSPOOL request to retrieve this response from the LST queue.

MODE=SPOOL is valid only if REQ=COMMAND is specified and the submitted command is "PDISPLAY queue" or "PDISPLAY PNET". The job name assigned to the queue entry by VSE/POWER is $SPLnnnn (where nnnn = the job number assigned by VSE/POWER). The queue entry's class and disposition are the ones contained in the SPL.

NEWVAL=value|(reg)

For value, specify the new value that is to be used by VSE/POWER as the job attribute. You can specify this value in one of the following forms:

```
C'x'    For example:  NEWVAL=C'A'
X'nn'   For example:  NEWVAL=X'F1'
n       For example:  NEWVAL=5
```

The operand is valid only together with one of the following specifications:

```
REQ=CLASS    For a new class of the job
REQ=DISP     For a new disposition of the job
REQ=PRI      For a new priority of the job
REQ=REMOTE   For a new remote ID
```

If you use a register, it must contain the new value.

PBUF=buffaddr|(reg)

The operand specifies the address of a buffer which is for use by VSE/POWER and for VSE/POWER feedback information. This buffer must be 88 bytes long.

If you use a register, it must contain the buffer's address.

PWD=password|(reg)

For password, specify the password for the VSE/POWER job or output.

If a password was defined on input or in an * $$ LST or * $$ PUN statement, then the same password is to be specified to have VSE/POWER execute any queue manipulation commands (such as PALTER or PDELETE). If there is no match of the passwords, then VSE/POWER rejects the request with a return code in the error/feedback bytes of the SPL for the request.

If you omit this operand (and also in the SPL macro) you will be given access also to a queue entry which was submitted without password protection via a local spool device.

If you use a register, it must point to an eight-byte field that contains the password left-justified.

QUEUE=LST|PUN|RDR|XMT

The operand specifies the queue to be used for the CTLSPOOL request:

```
LST      For list queue
PUN      For punch queue
RDR      For reader queue
XMT      For transmission queue
```

The operand is ignored if one of the following is specified

REQ=CANCEL

REQ=COMMAND

REQ=STATUS

REQ=...

The operand specifies the requested operation as follows:

CANCEL

Applies only to job input; causes the affected job to be deleted from the input queue if it has not yet been processed.

CLASS
Alters the job class of the job on the specified VSE/POWER queue. Requires a NEWVAL=value specification in order to be valid.

COMMAND
Indicates that you have supplied a VSE/POWER command in the area defined in the PBUF operand. No error detection is performed for the command, and no error code is returned, except for an invalid request (an invalid SPL address, for example). You must analyze the PBUF area in your program for a possible return message. Your program can pass only one of the following commands per CTLSPOOL request:

```
PALTER queue,jobname        (See Note 1 below)
PBRDCST
PCANCEL jobname             (See Note 1 below)
PDELETE queue,jobname       (See Note 1 below)
PDELETE MSG
PDISPLAY queue,jobname
PDISPLAY CRE
PDISPLAY DEL
PDISPLAY TOTAL
PDISPLAY BIGGEST  PDISPLAY MSG
PDISPLAY A
PDISPLAY T
PDISPLAY DYNC
PDISPLAY PNET
PHOLD queue,jobname         (See Note 1 below)
PINQUIRE
PLOAD DYNC                  (See Note 2 below)
PRELEASE queue,jobname      (See Note 1 below)
PVARY DYNC                  (See Note 2 below)
PXMIT
```

Note:

1. The command can be used in a networking environment for execution at another node if that other node is controlled by VSE/POWER.

   VSE/POWERpasses only one message back to your program in reply to any command.

   VSE/POWERprocesses the command on your own z/VSE node if both the user ID and the password match the explicit user ID and password defined for the job or its output.

   On another node controlled by VSE/POWER, the command is presented only if the user ID matches the one specified for the affected job or its output.
2. These commands are valid only for requests entitled by a hex zero password.

DISP
Alters the disposition of the affected queue entry. Requires a NEWVAL=value specification in order to be valid.

LOOKUP
Causes status information about the specified job or output to be returned in applicable fields of the SPL. VSE/POWER returns the following:

Job number

Class

Disposition

Number of lines or cards

Flag (indicating that more than one queue entry exists)

PRI
Alters the priority of the affected queue entry. Requires a NEWVAL=value specification in order to be valid.

REMOTE
Alters the remote ID to which output of the job is to be routed. Requires a NEWVAL=value specification in order to be valid.

SCRATCH
Causes the named job to be deleted from the affected VSE/POWER output (LST, PUN, or XMT) queue.

STATUS
Causes the following to be passed to the named SPL:

1. The disposition of the named job in the field SPQD of the SPL.
2. The job's queue indicator in the field SPSQ of the SPL. This indicator may be:

```
L =   The job is in the LST queue.
N =   Nothing to display (the specified job name is unknown).
P =   The job is in the PUN queue.
R =   The job is in the RDR queue.
X =   The job is in the XMT queue.
```

If the job exists in several queues, only its first occurrence is returned. The queues are searched in this sequence: LST, RDR, PUN, XMT.

(reg)
Indicates that a request code is provided in the specified register. You can specify one of the following codes in this register:

Code	Request Type	Requested Function	Corresponding Command
X'01'	PRI	Alter the priority	PALTER
X'02'	DISP	Alter the disposition	PALTER
X'04'	CLASS	Alter the class	PALTER
X'08'	REMOTE	Alter the remote identifier	PALTER
X'10'	CANCEL	Cancel input	PDELETE RDR
X'20'	SCRATCH	Scratch output	PDELETE queue
X'40'	STATUS	Display the status of the named job	PDISPLAY
X'80'	COMMAND	Process the passed VSE/POWER command	-

USERID=user-id|(reg)
For user-id specify an alphameric string of up to eight characters. For user-id, specify the explicit user ID of the queue entry that is to be manipulated. This ID was defined when the job was submitted to VSE/POWER or read in locally. If you omit the operand, VSE/POWER uses, for your CTLSPOOL request, the user ID currently stored in the request SPL.

VSE/POWER rejects your request if:

- You specified an ID which does not match the originally defined one.
- You did not specify an ID, but the ID currently stored in the SPL does not match the originally defined one.
- You did not specify an ID, no ID is stored in the SPL, and an explicit ID was defined for the affected queue entry.

If you use a register, it must point to an eight-byte field that contains the ID left-justified.

GETSPOOL Macro: Retrieve Data from Queues

The macro requests the retrieval of data currently held in VSE/POWER queues on disk. VSE/POWER returns the requested data to the buffer area of the partition issuing the GETSPOOL macro.

VSE/POWER accepts the request only if the affected queue entry's disposition is D or K. As for an output task, the entry's disposition is changed to L after processing if this disposition was K, the entry is deleted if this disposition was D. Therefore, before you can retrieve a queue entry processed by VSE/POWER previously, you must issue a CTLSPOOL request that changes this entry's disposition from L to K again.

If you use GETSPOOL and do not read to the end of data, a problem can occur. The accessed queue entry remains in the VSE/POWER queue in an active state and the operator cannot delete the entry (VSE/POWER displays DISP=*). You can avoid this by one of the following actions:

- Delete the entry using a CTLSPOOL request.
- Submit a CTLSPOOL request following your GETSPOOL, for example:
  ```
  CTLSPOOL SPL=(reg),REQ=STATUS
  ```
- Always read a queue entry until end-of-data.
- Request a GETSPOOL operation for another queue entry.

Any of these actions causes the entry to be deleted (disposition was D) or closed and retained with disposition L (disposition was K).

In response to the first GETSPOOL request, VSE/POWER returns, in your SPL, the number of records which the entry contains.

When end of data is reached, VSE/POWER returns the EOF indicator as a dummy record after the last data record. With buffered GETSPOOL requests, VSE/POWER returns the EOF indicator in the prefix of the last data record.

Requirements For the Caller

AMODE:
24

RMODE:
24

ASC Mode:
Primary

Format of the Macro

```
                                     ┌─,CC=NO──┐
►►──┬──────┬──GETSPOOL SPL=(reg)─────┼─────────┼──┬────────────────────────┬──────────────────►
    └─name─┘                         └─,CC=YES─┘  └─,CLASS=──┬─class─┬────┘
                                                             └─(reg)─┘

►──┬──────────────────────┬──┬─────────────────────────┬──┬─────────────────────────┬────────►
   └─,JNUM=──┬─SPL───┬────┘  └─,JOBN=──┬─jobname─┬─────┘  └─,LINENO=──┬─number─┬───┘
             └─(reg)─┘                 └─(reg)───┘                    └─(reg)──┘

►──┬─────────────┬──┬──────────────────────────┬──┬──────────────────────────────┬──────────►
   └─,MODE=BUF───┘  └─,PBUF=──┬─buffaddr─┬─────┘  └─,PBUFL=──┬─bufflength─┬─────┘
                              └─(reg)────┘                   └─(reg)──────┘

                                    ┌─,QUEUE=LST─┐
►──┬───────────────────────────┬────┼────────────┼──┬──────────────────────────┬────────────►◄
   └─,PWD=──┬─password─┬───────┘    └─,QUEUE=PUN─┘  └─,USERID=──┬─user-id─┬───┘
            └─(reg)────┘                                         └─(reg)───┘
```

SPL=(reg)
This mandatory operand specifies the register which contains the address of the SPL to be used. The SPL defines the request to VSE/POWER.

If you used the LINENO operand in a preceding GETSPOOL request, specify the address of the same SPL in this request; else, line positioning gets lost.

CC=YES|NO
Specify CC=YES to have VSE/POWER return the command code of the CCW for the currently processed data record. VSE/POWER inserts this code in the field SPCC of the SPL, except when you specify also MODE=BUF.

If you specify also MODE=BUF, VSE/POWER passes all command codes, including those which have no associated data, to your program's buffer.

CLASS=class|(reg)
For class, specify the class value assigned to the queue entry (by PUTSPOOL, for example). If you use a register, supply the applicable class value in it.

JNUM=SPL|(reg)
Use this operand if several jobs in the accessed queue have the same name.

Specify JNUM=SPL if VSE/POWER is to use the job number currently stored in the SPL. Supply the VSE/POWER-assigned job number in the specified register otherwise. If you omit the operand, VSE/POWER sets the SPL's job number field to zero.

JOBN=jobname|(reg)
For jobname, specify the name by which VSE/POWER knows the queue entry that is to be retrieved. It is the name that was assigned to the entry (by PUTSPOOL, for example).

If you use register notation, the specified register must point to an eight-byte field containing the name in that field left-adjusted.

LINENO=number|(reg)
Use this operand in your first GETSPOOL request for a queue entry if retrieval

is to begin with a certain output record. The operand causes retrieval to begin at the specified line number relative to the beginning of the file. If you use a register, supply the line number in it.

The maximum value that you can specify for number is 16777215.

If you omit the operand, VSE/POWER starts retrieval at the beginning of the queue entry's spool data.

To read records in consecutive order, omit this operand in a second or subsequent GETSPOOL request to the same queue entry. For random retrieval, specify the operand in subsequent GETSPOOL requests for repositioning VSE/POWER's line pointer accordingly. Ensure, however, that the subsequent GETSPOOL requests use the same SPL as the initial request for this retrieval operation.

MODE=BUF

Specify this operand if VSE/POWER is to retrieve more than one record per request. The operand causes VSE/POWER to fill the area named in the PBUF operand with as many records as will fit. Every data record in that area has a four-byte prefix as follows:

Byte	Contents
0	Command code. If VSE/POWER is to pass also command-code-only records (such as a skip to channel 1), you must specify CC=YES in addition.
1	X'80' = The last record in the buffer. X'C0' = The last record of the spool data.
2-3	Length (in binary) of a data record, including the four-byte prefix.

Deblocking is to be done in your program.

PBUF=buffaddr|(reg)

For buffaddr, specify the symbolic address of the buffer into which VSE/POWER is to pass retrieved data records or feedback information (on certain error conditions) or both. If you use this operand, you must also specify PBUFL=bufflength.

You can omit this operand and the PBUFL operand if you defined a buffer in your SPL.

If you use register notation, the specified register must point to the buffer which VSE/POWER is to use.

PBUFL=bufflength|(reg)

The operand specifies the length (in number of bytes) of the buffer whose address is given in the PBUF operand. Define your buffer's length large enough for your longest data record to fit into the buffer. VSE/POWER truncates the trailing blanks of a record; it indicates the length of every record after truncation in either:

- The four-byte SPL field SPRL if data records are not blocked, or
- Bytes 2 and 3 of the record prefix if the data records are blocked (see also the MODE=BUF operand).

The minimum length you can specify is 88.

If you use a register, it must contain the buffer's length.

PWD=password|(reg)
For password, specify the explicit password for the queue entry to be retrieved. If there is no match of the passwords, VSE/POWER rejects the request with a return code in the error/feedback bytes of the SPL for the request.

If you omit this operand (and also in the SPL macro), you will be given access also to a queue entry which was submitted without password protection via a local spool device.

If you use register notation, the specified register must point to an eight-byte field that contains the password left-justified.

QUEUE=LST|PUN
The operand specifies the queue to which the GETSPOOL request applies:

LST
For list queue

PUN
For punch queue

USERID=user-id|(reg)
For user-id specify an alphameric string of up to eight characters. For user-id, specify the user ID associated with the queue entry to be retrieved. This ID was defined when the job was submitted to VSE/POWER. If you omit the operand, VSE/POWER uses, for your GETSPOOL request, the ID currently stored in the request SPL.

If you use a register, it must point to an eight-byte field that contains the ID left-justified.

Note: Currently this user-id value is not used by VSE/POWER when checking correct access!

PUTSPOOL Macro: Submitting a Job Stream

You use the macro to submit a job stream from your program's buffer to the:

- VSE/POWER RDR queue for later execution in a partition under control of VSE/POWER
- VSE/POWER transmission (XMT) queue for transmission to an other node

VSE/POWER analyses only those JECL statements which you submit with the first PUTSPOOL request for a queue entry. VSE/POWER places these statements into the input queue. For example, if you wish to specify output characteristics (such as class or disposition) other than the default values, supply an * $$ LST or * $$ PUN statement.

If your program does not pass an * $$ JOB statement, VSE/POWER builds this statement (in accordance with your specifications for the applicable SPL) and inserts it into your job stream.

For the second and subsequent PUTSPOOL requests, VSE/POWER passes your input from the buffer to the VSE/POWER input queue. No more checking is performed. When the last statement of the input has been read from the buffer and no more continuation input exists, VSE/POWER inserts an * $$ EOJ statement if one has not been passed.

The job number assigned by VSE/POWER is returned to your program in the job-number field of the SPL. You may want to use this job number later together with the job name in order to retrieve the job's output.

If there is a user-written JOBEXIT routine for local input, VSE/POWER passes to this routine the z/VSE job-control statements and JECL statements of the submitted jobs.

Requirements For the Caller

AMODE:
24

RMODE:
24

ASC Mode:
Primary

Format of the Macro

```
>>─┬──────┬─PUTSPOOL SPL=(reg)─┬──────────────────────────┬──────────────>
   └─name─┘                    └─,CBUF=─┬─firstbuffaddr─┬─┘
                                        └─(reg)─────────┘

>─┬──────────────┬─┬─────────────────────┬─┬─────────────────────┬─────>
  └─,CONT=(reg)──┘ └─,JOBN=─┬─jobname─┬──┘ └─,PBUF=─┬─buffaddr─┬─┘
                            └─(reg)───┘             └─(reg)────┘

>─┬─────────────────────┬─┬─────────────────────────┬──────────────────><
  └─,PWD=─┬─password─┬──┘ └─,USERID=─┬─user-id─┬────┘
          └─(reg)────┘               └─(reg)───┘
```

SPL=(reg)
For reg, specify the register that contains the address of the SPL that is to be used by the PUTSPOOL macro. The SPL defines the request to VSE/POWER.

CBUF=firstbuffaddr|(reg)
For firstbuffaddr, specify the symbolic address of the first of the 88-byte buffers that contain the job stream. The format of an 88-byte buffer is as follows:

Bytes	Contents
0-3	A pointer to the next buffer in the chain (0 for the last buffer)
4-7	Reserved
8-87	An 80-byte data buffer data

If the CONT operand is specified together with CBUF, the same buffers must be reused for the continuation data. If register notation is used, the continuation routine must reset the previously used register contents every time continuation data is made available. See also the description of the CONT operand below.

The CBUF pointer in the SPCB field of the SPL is used as a work area and is set to zero when PUTSPOOL processing is ended. If the PUTSPOOL macro uses an SPL which has already been used, or if the PUTSPOOL macro is entered more than once with the same SPL, one of the following is required:

- CBUF is specified to reset the field SPCB of the SPL.
- The field SPCB of the SPL is updated (a maximum of 4,095 input buffers is allowed for every PUTSPOOL access).
- The buffer defined by CBUF is the only one (its chain pointer is zero).

CONT=(reg)
If the buffers processed by this execution of PUTSPOOL do not contain the complete job stream, this operand should be used to give the address of a continuation routine. In this routine, you can submit further data buffers associated with the same job stream. However, no other operands can be changed in the continuation routine.

When this operand is used, then also the CBUF operand must be specified.

To exit from the routine, set the specified register to zero and return to the PUTSPOOL macro or, via register 14, to VSE/POWER.

JOBN=jobname|(reg)
For jobname, specify the (unique) name that is to be assigned to the job in the VSE/POWER input queue. This name is to be used if, for example, the job's output is to be retrieved by a GETSPOOL macro or if the job is to be accessed by a CTLSPOOL macro.

PBUF=buffaddr|(reg)
For buffaddr, specify the address of a buffer for use by VSE/POWER and for VSE/POWER feedback information on certain error conditions. The size of this buffer must be at least 88 bytes. If register notation is used, the specified register must contain a pointer to this buffer.

PWD=password|(reg)
Use this operand to define the password for this VSE/POWER job.

A password which you specify in a PUTSPOOL macro:

- Overrides the password that may be stored in the request SPL.
- Must be used in any subsequent GETSPOOL and CTLSPOOL macro for the job.
- Can be overridden for output by the PWD operand of an * $$ LST statement or * $$ PUN statement.

The password can be any string of up to eight alphameric characters.

USERID=user-id|(reg)
For user-id, specify the owning user-ID which is to be associated with the queue entry that is to be placed into one of the VSE/POWER queues (RDR or XMIT).

If you use register notation, the specified register must point to an eight-byte field containing the ID left-justified.

If you omit this operand and do not supply a user ID with the SPL macro defining the request SPL, VSE/POWER spools the applicable job input with a default blank user ID.

Return Codes for CTLSPOOL, GETSPOOL, and PUTSPOOL

When issuing a PUTSPOOL, GETSPOOL, or CTLSPOOL macro, the return codes are issued in the SPL and/or in register 15.

Return Codes in Register 15

VSE/POWER's SPOOL macro support makes use of the z/VSE macros XPOST and XWAIT; your program should examine their return codes in register 15. For a description of these macros and their return codes, see *z/VSE System Macros Reference*, SC34-2638. In order to distinguish between return codes provided by XPOST and XWAIT, XPOST return codes are multiplied by 16. This means bit 24 to bit 27 of register 15 contain the return code for XPOST. Bit 28 to bit 31 contain the return code for XWAIT. If XPOST sets any return code, XWAIT will not be processed.

Register 15 contains the code X'40' if VSE/POWER control tables were generated without SPOOL=YES specified in the VSE/POWER generation macro.

Register 15 contains the code X'C0' if VSE/POWER is running in a partition allocated in a private address space and the user program is running in a partition allocated in a different private address space (and no other error occurred). In this case, either allocate the VSE/POWER partition in the shared address space or let the user program run in a partition allocated in the same address space as the VSE/POWER partition.

Return Codes in the SPL

Table 83 on page 360 shows the return codes that your program receives following the processing of a PUTSPOOL, GETSPOOL, or CTLSPOOL macro. VSE/POWER supplies these codes as follows:

- In the SPL bytes SPER and SPER2
- In a byte you can access using the DSECT generated by SPL TYPE=MAP,...

  You access this byte by referring to field xxxXECB+4, where xxx is either SPM or ICR depending on the type of SPL.

Additional information is passed to your program in error-feedback byte 2 (SPER2) of the SPL:

Mnemonic of Equate	Hex. Value	Meaning
SPAI	80	Wrong password - access denied
SPAC	40	Job/Output Spool Access Protection violation. VSE/POWER has been started with Spool Access Protection active, and the given spool entry does not specify SECAC=NO, and an XECB program attempted to access a spool entry. However, either: - the program's security logon userid (from either the IBM Component terminal logon or the partition from the // ID or * $$ JOB SEC= statement) does not match the spool entry's authorized access userid(s) (either the spool entry's origin or target userid), or - the spool entry specifies a target userid of 'ANY' and the program does not have a security logon userid

Mnemonic of Equate	Hex. Value	Meaning
		The authorized access userid(s) can be displayed with the PDISPLAY command (displayed as FROM= or TO=).
SPDDR	02	3540 data-mode record is being processed

Table 83. Return Codes Supplied in the SPL

Return Code	Meaning	Passed in xxxXECB +4	Passed in SPER
X'0x'	*Miscellaneous*:		
X'08'	End of data encountered during a GETSPOOL request, or invalid LINENO specified in GETSPOOL.	X	X
X'09'	Task was waiting on queue/account file space.		
X'1x'	*Invalid specification*:		
X'11'	Command not allowed.	X	X
X'12'	Invalid VSE/POWER output disposition in SPL.	X	X
X'14'	Invalid output class (not A-Z, not 0-9) in SPL.	X	X
X'15'	Invalid user Id (not A-Z) in SPL.	X	X
X'16'	Invalid queue specified.	X	X
X'17'	Invalid password specified.	X	X
X'18'	Invalid job name in SPL.	X	X
X'2x'	*Job processing errors*:		
X'21'	The PBUF buffer area is smaller than 88 bytes or not large enough to hold the largest output data record (PBUFL in GETSPOOL too small).	X	X
X'22'	GETSPOOL was unable to locate output file by specified job name, job class, and dispatch-able VSE/POWER disposition, or requested output file is in use. If an invalid password was specified, X'80' appears in field SPER2 of the SPL also.	X	X
X'24'	A loop occurred in the PUTSPOOL buffer chain, or more than 4095 buffers were used per request.	X	X
X'28'	Invalid CTLSPOOL REQ operand.	X	X
X'4x'	*VSE/POWER diagnostic*:		
X'41'	VSE/POWER terminated normally, or VSE/POWER terminated abnormally, or VSE/POWER spool management task terminated abnormally.	X X X	X
X'42'	A VSE/POWER message was logged during CTLSPOOL (see Notes below).	X	X
X'44'	A VSE/POWER error occurred during GETSPOOL (see Notes below).	X	X
X'48'	A VSE/POWER error occurred during PUTSPOOL (see Notes below).	X	
X'49'	Task waiting on queue/account file space. This return code will not appear when the PUTSPOOL/ GETSPOOL user receives control back fromVSE/POWER .		
X'8x'	*Invalid address pointer*:		X
X'82'	Invalid data buffer chain (PUTSPOOL)	X	X
X'84'	Invalid VSE/POWER buffer address (PBUF)	X	
X'88'	Invalid SPL address	X	

Table 83. Return Codes Supplied in the SPL (continued)

Return Code	Meaning	Passed in xxxXECB +4	Passed in SPER
Notes: 1. The first 60 characters of the VSE/POWER message are displayed at displacement 28 of the buffer defined by the PBUF operand. 2. No spool management error detection is done for a CTLSPOOL with REQ=COMMAND. Your program must analyze the message returned by VSE/POWER in the buffer defined by the PBUF operand. If the command passed by a CTLSPOOL request results in more than one message, VSE/POWER returns only the message that best describes the condition. 3. All values specified in the NEWVAL operand of CTLSPOOL must conform to the related VSE/POWER rules.			

Coding Example for Using the SPOOL Macros

Figure 10 gives a coding example for the use of the SPOOL macro support. Figure 11 on page 369 shows the console listing that resulted from running the example, and Figure 12 on page 370 shows the corresponding list output.

The example submits a job made up of numbered card-image records. The output of the job is retrieved, with GETSPOOL, both sequentially and randomly. The output is displayed at the same time.

In the example, the F2 partition is used by VSE/POWER, the F3 partition processes the reader CLASS=A input, and the job runs in the BG partition.

Columns 119 and 120 of Figure 12 on page 370 contain the CCW command code on account of the CC=YES specification in the GETSPOOL macro.

Figure 10. Coding Example for the Use of SPOOL Macros

```
         PUNCH   '     PHASE EXAMPLE,S'
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                     *
*                       V S E / P O W E R                             *
*                                                                     *
*             C R O S S - P A R T I T I O N   E X A M P L E           *
*                                                                     *
*                                                                     *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         SPACE 3
* THE FOLLOWING IS AN EXAMPLE OF THE USAGE OF VSE/POWER CROSS-PARTITION
* SUPPORT. IT CONFORMS TO THE SUGGESTED PROGRAMMING PRACTICES SO AS
* TO ALLOW A MAXIMUM OF FRICTION-FREE EXISTENCE BETWEEN MULTIPLE USERS.
*
* REGISTER USAGE:
*
*        R0  - VSE WORK REG (SETIME MACRO)
*        R1  - WORK REGISTER,DOS/VSE WORK REG (XECBTAB MACRO)
*        R2  - WORK REGISTER
*        R3  - WORK REGISTER
*        R4  - WORK REGISTER
*        R5  - PNTR TO PUTSPOOL CARD INPUT AREA,WORK REGISTER
```

```
*        R6  - LINK RETURN REG: PUT,TESTSTAT,HEXCONV ROUTINES
*        R7  - BASE REG 2
*        R8  - GETSPOOL LINENO= PARAMETER VALUE
*        R9  - BASE REG 1
*        R10 - LINK RETURN REG: GETSUB,XRETRY ROUTINE
*        R11 - SPL PNTR
*        R12 - PUTSPOOL CONT= PNTR
*        R13 -
*        R14 - VSE WORK REG (XECBTAB MACRO)
*        R15 - VSE WORK REG
         SPACE 1
R0   EQU    0
R1   EQU    1
R2   EQU    2
R3   EQU    3
R4   EQU    4
R5   EQU    5
R6   EQU    6
R7   EQU    7
R8   EQU    8
R9   EQU    9
R10  EQU    10
R11  EQU    11
R12  EQU    12
R13  EQU    13
R14  EQU    14
R15  EQU    15
         SPACE 3
         EJECT
         PRINT NOGEN
         CSECT
         BALR  R9,0            ESTABLISH ADDRESSABILITY
         USING *,R9,R7          LA    R7,4095(,R9)
         LA    R7,1(,R7)
         SPACE
         OPEN SYSLST
         SPACE
         MVC   LINE,=CL120'EXAMPLE BEGIN'
         BAL   R6,PUT          PRINT START OF EXEC MSG
         SPACE
***********************************************************************
*        PUTSPOOL SECTION                                             *
***********************************************************************
         SPACE
* THE PUTSPOOL XECB IS DEFINED AND IF DEFINE IS NOT SUCCESSFUL,
* THEN A RETRY WITH A COUNTER IS MADE. IF NOT SUCCESSFUL, THEN
* EITHER ANOTHER USER HAS IT RESERVED, OR THE VSE XECB
* TABLE IS FULL.
         SPACE
         XR    R2,R2           SET RETRY COUNTER
         MVC   LINE,ERRMSG0    INIT MSG AREA
XDEFPUT  XECBTAB TYPE=DEFINE,XECB=ICRXECB,ACCESS=XWAIT
         LTR   R15,R15         ERROR RETURN?
         BZ    PUTSPOOL        NO
         SPACE
* XECBTAB ERROR RETURN
         LA    R10,XDEFPUT      LOAD RETRY RETURN PNTR
         B     XRETRY          RETRY
         SPACE
* THE PUTSPOOL XECB IS NOW OWNED.
* THE POINTER REGS ARE LOADED FOR THE PUTSPOOL CALL.
PUTSPOOL MVC   LINE,=CL120'EXAMPLE PUTSPOOL:'
         BAL   R6,PUT           PRINT HEADER
         SPACE
         LA    R11,SPLEX        LOAD PUTSPOOL SPL=PNTR
         USING SPL,R11
         ST    R11,ICRXECB+4    INITIALIZE XECB
```

```
         LA    R12,PUTCONT      LOAD PUTSPOOL CONT= PNTR
         LA    R5,CARDS1        LOAD PUTSPOOL CBUF= PNTR
         SPACE
PUTLOOP  PUTSPOOL SPL=(R11),CONT=(R12),CBUF=(R5),JOBN=EXAMPLE
         SPACE
* DO ERROR CHECKING FOLLOWING PUTSPOOL.
         MVC   LINE,ERRMSG1   INIT MSG AREA
         LTR   R15,R15        VSE ERROR RTN ?
         BNZ   ERRX           YES         MVC   LINE,ERRMSG2   INIT MSG AREA
         LA    R2,ICRXECB     INIT PNTR FOR PUTSPOOL ERR RTN CHECK
         BAL   R6,TESTSTAT    CHECK FOR PUTSPOOL ERROR RTN
         SPACE
* DELETE THE PUTSPOOL XECB NOW TO ALLOW OTHER USERS ACCESS
         XECBTAB TYPE=DELETE,XECB=ICRXECB
         B     JOBWAIT
         SPACE 3
***********************************************************************
*        PUTSPOOL CONTINUATION ROUTINE
***********************************************************************
         SPACE
* DO ERROR CHECKING
PUTCONT  MVC   LINE,ERRMSG1   INIT MSG AREA
         LTR   R15,R15        VSE ERROR RTN?
         BNZ   ERRX           YES
         MVC   LINE,ERRMSG2   INIT MSG AREA
         LA    R2,ICRXECB     LOAD PNTR FOR PUTSPOOL ERR RTN CHECK
         BAL   R6,TESTSTAT    CHECK FOR PUTSPOOL ERROR RTN
         SPACE
* SET UP FOR PUTSPOOL CONTINUATION AND RETURN TO PUTSPOOL.
         LA    R5,CARDS2      LOAD PNTR TO NEXT INPUT
         LA    R12,0          INDICATE END OF INPUT
         B     PUTLOOP        RETURN TO PUTSPOOL
         SPACE 3
***********************************************************************
*        CHECK FOR JOB COMPLETION
***********************************************************************
* A WAIT WITH TIMER INTERRUPT IS SCHEDULED IN ORDER TO ALLOW ANY
* PARTITIONS WITH A LOWER PRIORITY TO EXECUTE WHILE WAITING ON
* PUTSPOOL INPUT TO EXECUTE. THIS IS ESPECIALLY
* IMPORTANT IF THIS PARTITION HAS A HIGHER PRIORITY THAN THE
* VSE/POWER PARTITION!!
         SPACE
JOBWAIT  LA    R11,SPLEX
         ST    R11,SPMXECB+4   INIT XECB
         SPACE
CTLLOOP  SETIME 1,TECB        SET TIMER INTERRUPT
         WAIT  TECB
         SPACE
* DEFINE CTL/GETSPOOL XECB
         XR    R2,R2          INIT RETRY COUNTER
XDEFCTL  XECBTAB TYPE=DEFINE,XECB=SPMXECB,ACCESS=XWAIT
         LTR   R15,R15        ERROR RTN?
         BZ    CTLSP1         NO
         SPACE
* XECBTAB ERROR RETURN
         MVC   LINE,ERRMSG3   INIT MSG AREA
         LA    R10,XDEFCTL    LOAD PNTR FOR XRETRY
         B     XRETRY
         SPACE
* XECB IS NOW OWNED AFTER TIMER PAUSE. NOW PROCEED WITH CTLSPOOL CALL.
CTLSP1   CTLSPOOL SPL=(R11),REQ=STATUS
         SPACE
* CHECK FOR ERROR
         MVC   LINE,ERRMSG4   INIT MSG AREA
         LTR   R15,R15        VSE ERROR RTN?
         BNZ   ERRX           YES
         MVC   LINE,ERRMSG5   INIT MSG AREA
```

```
         LA    R2,SPMXECB     LOAD PNTR FOR TESTSTAT
         BAL   R6,TESTSTAT    CHECK FOR CTLSPOOL ERROR RTN
         SPACE
         CLI   SPSQ,C'R'      JOB STILL IN RDR QUEUE?
         BE    CTLDEL         YES,DELETE XECB AND LOOP
         CLI   SPSQ,C'L'      IS JOB IN THE LST QUEUE?
         BE    GETSPOOL       YES, KEEP XECB AND DO GETSPOOL
         MVC   LINE,ERRMSGX   ERROR MSG AREA, INIT MSG AREA
         BAL   R6,PUT
         SPACE
* DELETE XECB AND EXIT
         XECBTAB TYPE=DELETE,XECB=SPMXECB
         EOJ
         SPACE
* DELETE XECB AND RETRY AFTER TIMER WAIT
CTLDEL   XECBTAB TYPE=DELETE,XECB=SPMXECB DELETE CTL/GETSPOOL XECB
         B     CTLLOOP        LOOP
         EJECT
***********************************************************************
*        GETSPOOL SECTION - SEQUENTIAL RETRIEVAL
***********************************************************************
         SPACE
GETSPOOL MVC   LINE,=CL120'EXAMPLE GETSPOOL SEQUENTIAL:'
         BAL   R6,PUT
         SPACE
GETLOOP  XC    PBUF,PBUF      CLEAR OUTPUT BUFFER
         SPACE
         GETSPOOL SPL=(R11),CC=YES    RETRIEVE WITH CMND CODES
         SPACE
* CHECK FOR ERROR
         MVC   LINE,ERRMSG6   INIT MSG AREA
         LTR   R15,R15        VSE ERROR RTN?
         BNZ   ERRX           YES
         MVC   LINE,ERRMSG7   INIT MSG AREA
         LA    R2,SPMXECB     LOAD PNTR FOR TESTSTAT
         BAL   R6,TESTSTAT    CHECK FOR GETSPOOL ERROR RTN
         SPACE
* PRINT OUT RECORD RETRIEVED WITH COMMAND CODE.
         MVC   LINE,PBUF      MOVE RETRIEVED OUTPUT TO PRINT BUF
         XR    R3,R3
         IC    R3,SPCC        LOAD LST RECORD CMND CODE
         LA    R4,CC
         BAL   R6,HEXCONV     CONVERT CMND CODE TO EBCDIC
         BAL   R6,PUT         PRINT OUTPUT REC
         SPACE
* TEST FOR END OF OUTPUT
         TM    SPER,SPLR      END OF DATA?
         BNO   GETLOOP        NO, LOOP BACK AND GET NEXT RECORD
         SPACE 3
***********************************************************************
*        GETSPOOL SECTION - BROWSING (RANDOM RETRIEVAL)
***********************************************************************
         SPACE
* JOB NOW HAS DISP=L AFTER RETRIEVAL. CHANGE BACK TO DISP=K IN ORDER
* TO RETRIEVE AGAIN.
         SPACE
         CTLSPOOL SPL=(R11),REQ=DISP,NEWVAL=C'K'
         SPACE
* CHECK FOR ERROR
         MVC   LINE,ERRMSG8   INIT MSG AREA
         LTR   R15,R15        VSE ERROR RTN?
         BNZ   ERRX           YES        MVC   LINE,ERRMSG9   INIT MSG AREA
         LA    R2,SPMXECB     LOAD PNTR FOR TESTSTAT
         BAL   R6,TESTSTAT    CHECK FOR CTLSPOOL ERROR RTN
         SPACE
* PRINT OUT HEADER
         MVC   LINE,=CL120'EXAMPLE GETSPOOL RANDOM RETRIEVAL:'
```

```
         BAL   R6,PUT
         SPACE
* GET LINE 3
         LA    R8,3           LOAD LINENO VALUE
         BAL   R10,GETSUB     CALL GETSPOOL AND PRINT LINE
         SPACE
* GET LINE 2
         LA    R8,2           LOAD LINENO VALUE
         BAL   R10,GETSUB     CALL GETSPOOL AND PRINT LINE
         SPACE
* GET LINE 4
         LA    R8,4           LOAD LINENO VALUE
         BAL   R10,GETSUB     CALL GETSPOOL AND PRINT LINE
         SPACE 3
***********************************************************************
*        DELETE OUTPUT AND EXIT
***********************************************************************
         SPACE
         CTLSPOOL SPL=(R11),REQ=SCRATCH
         SPACE
* CHECK FOR ERROR
         MVC   LINE,ERRMSG10  INIT MSG AREA
         LTR   R15,R15        VSE ERROR RTN?
         BNZ   ERRX           YES
         MVC   LINE,ERRMSG11  INIT MSG AREA
         LA    R2,SPMXECB     LOAD PNTR FOR TESTSTAT
         BAL   R6,TESTSTAT    CHECK CTLSPOOL ERROR RTN
         SPACE 3
         MVC   LINE,=CL120'EXAMPLE SUCCESSFUL'
         BAL   R6,PUT
         SPACE
ERREND   DS    0H
* EXIT - XECB'S DEFINED AT THIS TIME ARE DELETED BY VSE AT EOJ, SO
*     NO XECBTAB TYPE=DELETE IS NECESSARY
         EOJ
         SPACE 3
         EJECT
***********************************************************************
*        GETSPOOL SUBROUTINE
***********************************************************************
* SUBROUTINE TO DO RANDOM GETSPOOL.
* INPUT REGS:
*        R8 - LINENO VALUE
*        R10 - LINK REG
*        R11 - SPL PNTR
         SPACE
GETSUB   XC    PBUF,PBUF      CLEAR GETSPOOL OUTPUT AREA
         GETSPOOL SPL=(R11),LINENO=(R8)
         SPACE* CHECK FOR ERROR
         MVC   LINE,ERRMSG12  INIT MSG AREA
         LTR   R15,R15        VSE ERROR RTN?
         BNZ   ERRX           YES
         MVC   LINE,ERRMSG13  INIT MSG AREA
         LA    R2,SPMXECB     LOAD PNTR FOR TESTSTAT
         BAL   R6,TESTSTAT    CHECK FOR GETSPOOL ERROR RTN
         SPACE
* PRINT RETRIEVED LINE
         MVC   LINE,=CL120'LINE XX ='
         LR    R3,R8
         LA    R4,LINE+5
         BAL   R6,HEXCONV     CONVERT LINE NUMBER TO EBCDIC
         BAL   R6,PUT         PRINT LINE NUMBER
         SPACE
         MVC   LINE,PBUF
         BAL   R6,PUT         PRINT PBUF
         SPACE
         BR    R10            RETURN
```

```
         SPACE 5
***********************************************************************
*        EXIT ROUTINE FOR VSE ERROR RTN HANDLING
***********************************************************************
         SPACE
ERRX     LR    R2,R15          LOAD VSE ERROR RTN CODE TO R2
         BAL   R6,PUT          PRINT MSG AREA
         MVC   LINE,=CL120'   ERROR RTN CODE IN REGISTER 2'
         BAL   R6,PUT
*        DUMP                  DUMP
         B     ERREND          EXIT
         EJECT
***********************************************************************
*        TESTSTAT SUBROUTINE - TESTS THE VSE/POWER RETURN CODE
***********************************************************************
* INPUT REGS:
*        R2 - ADDR OF CORRESPONDING VSE/POWER XECB
*        R6 - LINK REG
         SPACE
TESTSTAT TM    4(R2),SPIA+SPPP+SPUE+SPPI     ERROR RETURN ?
         BZR   R6              NO
         SPACE
* PRINT PREPARED MESSAGE WITH RTN CODE
         XR    R3,R3
         IC    R3,4(R2)        LOAD RETURN CODE
         LA    R4,RTNCODE      POINT TO HEXCONV OUTPUT AREA
         BAL   R6,HEXCONV      CONVERT RTN CODE TO HEX
         BAL   R6,PUT          PRINT OUT PREPARED MESSAGE AREA
         SPACE
* PRINT PBUF FOR POSSIBLE MESSAGE FROM VSE/POWER
         MVC   LINE,=CL120'PBUF='
         BAL   R6,PUT
         MVC   LINE,PBUF
         BAL   R6,PUT
         SPACE
*        DUMP         B     ERREND
         SPACE 5
***********************************************************************
*        HEXCONV SUBROUTINE - CONVERTS SINGLE BYTE TO TWO EBCDIC BYTES
***********************************************************************
* INPUT REGS:
*        R3 = INPUT BYTE TO CONVERT
*        R4 = PNTR TO OUTPUT AREA (TWO BYTES LONG)
*        R6 = LINK REG
HEXTBL   DC    C'0123456789ABCDEF' HEX CONVERT TABLE
         SPACE
HEXCONV  SLDL  R2,28           SHIFT LEFT HALF-BYTE TO R2 LOW ORDER
         STC   R2,0(R4)        STORE LEFT HALF-BYTE TO OUTPUT + 0
         SRL   R3,28           SHIFT RIGHT HALF-BYTE TO R3 LOW ORDER
         STC   R3,1(R4)        STORE RIGHT HALF-BYTE TO OUTPUT + 1
         TR    0(2,R4),HEXTBL TRANSLATE OUTPUT
         BR    R6              RETURN
         EJECT
***********************************************************************
*        XRETRY SUBROUTINE - RETRYS BLOCKED XECBTAB MACRO
***********************************************************************
* PRINTS A WARNING MESSAGE EVERY 16 SEC.
* INPUT REGS:
*        R2 - RETRY COUNTER (BEGINNING WITH ZERO)
*        R6 - RETURN REG
         SPACE
XRETRY   LA    R2,1(,R2)        INCREMENT COUNTER
         ST    R2,RETRYCNT     STORE FOR TRACING
         SPACE
* SET TIMER INTERRPT FOR 1 SEC.
         SETIME 1,TECB
         WAIT TECB
```

```
         SPACE
         C     R2,RETRYMAX     RETRY LIMIT EXCEEDED ?
         BH    XEND            YES, PRINT MSG AND EXIT
         SPACE
         TM    RETRYCNT+3,X'0F' RETRY COUNTER DIVISABLE BY 16?
         BNZR  R10             NO, RETRY WITHOUT WARNING MESSAGE
         SPACE
* PRINT INITIALIZED BUFFER MESSAGE
         BAL   R6,PUT          PRINT WARNING MESSAGE
         BR    R10             RETRY
         SPACE
* PRINT ERROR MESSAGE AND EXIT
XEND     MVC   LINE,ERRMSG14
         BAL   R6,PUT
         B     ERREND
         SPACE 5
***********************************************************************
*        PUT    SUBROUTINE - PRINTS LINE ON CONSOLE AND SYSLST
***********************************************************************
         SPACE
PUT      LA    R1,CCB
         EXCP  (R1)
         WAIT  (R1)          PUT   SYSLST
         BR    R6
         EJECT
***********************************************************************
*        CONSTANTS
***********************************************************************
         SPACE
* XECB'S
ICRXECB  DC    A(0,*-*)
SPMXECB  DC    A(0,*-*)
         SPACE
* ECB'S
TECB     TECB
         SPACE
* I/O BUFFERS
PBUF     DC    CL120' '
LINEX    DC    X'09'
LINE     DC    CL120' '
         ORG   LINE+118
CC       DS    2X
         ORG   ,
RTNCODE  EQU   LINE+39
         SPACE
* PUTSPOOL INPUT
CARDS1   DC    A(*+88,0)
         DC    CL80'// JOB XYZ'
         DC    A(0,0)
         DC    CL80'* CARD 2'
CARDS2   DC    A(*+88,0)
         DC    CL80'* CARD 3'
         DC    A(0,0)
         DC    CL80'/&&'
         SPACE
* RETRY VALUES
RETRYCNT DC    A(*-*)         CURRENT MAXIMUM RETRIES
RETRYMAX DC    F'600'         MAX RETRY - 10 MINUTES
         SPACE
* SPL MACRO'S
SPLEX    SPL   TYPE=DEFINE,PBUF=PBUF,PBUFL=120
         SPACE
* MESSAGES
ERRMSG0  DC   CL120'EXAMPLE WARNING:XECBTAB DEFINE OF ICRXECB BLOCKED'
ERRMSG1  DC   CL120'EXAMPLE ERROR:XPOST/XWAIT ERROR RTN FROM PUTSPOOL'
ERRMSG2  DC   CL120'EXAMPLE ERROR:PUTSPOOL  ERROR RTN CODE='
ERRMSG3  DC   CL120'EXAMPLE WARNING:XECBTAB DEFINE OF SPLXECB BLOCKED'
```

```
ERRMSG4  DC   CL120'EXAMPLE ERROR:XPOST/XWAIT ERROR RTN FROM CTLSPOOL1'
ERRMSG5  DC   CL120'EXAMPLE ERROR:CTLSPOOL1  ERROR RTN CODE='
ERRMSGX  DC   CL120'EXAMPLE ERROR:JOB LST OUTPUT NOT IN LST QUEUE'
*             (CAUSE IS EITHER ANOTHER TASK/X-PARTITION USER PROCESSED
*             THE OUTPUT, OR IT WAS NOT SPOOLED DURING EXECUTION)
ERRMSG6  DC   CL120'EXAMPLE ERROR:XPOST/XWAIT ERROR RTN FROM GETSPOOL1'
ERRMSG7  DC   CL120'EXAMPLE ERROR:GETSPOOL1 ERROR RTN CODE='
ERRMSG8  DC   CL120'EXAMPLE ERROR:XPOST/XWAIT ERROR RTN FROM CTLSPOOL2'
ERRMSG9  DC   CL120'EXAMPLE ERROR:CTLSPOOL2 ERROR RTN CODE='
ERRMSG10 DC   CL120'EXAMPLE ERROR:XPOST/XWAIT ERROR RTN FROM CTLSPOOL3'
ERRMSG11 DC   CL120'EXAMPLE ERROR:CTLSPOOL3 ERROR RTN CODE='
ERRMSG12 DC   CL120'EXAMPLE ERROR:XPOST/XWAIT ERROR RTN FROM GETSPOOL2'
ERRMSG13 DC   CL120'EXAMPLE ERROR:GETSPOOL2 ERROR RTN CODE='
ERRMSG14 DC   CL120'EXAMPLE ERROR:XECB DEFINE BLOCKED'
         SPACE 5
* I/O SECTION
CCB      CCB   SYSLOG,CCW
CCW      CCW   X'09',LINE,X'20',80
         SPACE
SYSLST   DTFDI DEVADDR=SYSLST,IOAREA1=LINEX,RECSIZE=121,MODNAME=MODNAME
         SPACE
MODNAME  DIMOD  TYPEFLE=OUTPUT
         EJECT
         PRINT GEN
* DSECT'S
SPL      SPL   TYPE=MAP
         CSECT
         LTORG
         END
```

```
BG
exec example
BG
EXAMPLE BEGIN
BG
EXAMPLE PUTSPOOL:
F2
1Q47I   F3 EXAMPLE 00020 FROM 000
F3
// JOB XYZ
DATE 07/11/78, CLOCK 09/04/51
F3
* CARD 2
F3
* CARD 3
F3
EOJ XYZ

DATE 07/11/78,CLOCK 09/04/54,DURATION 00/00/02
F2
1034I   F3 WAITING FOR WORK
BG
EXAMPLE GETSPOOL SEQUENTIAL:
BG

BG
// JOB XYZ
BG
* CARD 2
BG
* CARD 3
BG
EOJ XYZ
BG

BG

BG
EXAMPLE GETSPOOL RANDOM RETRIEVAL:
BG
LINE 03 =
BG
* CARD 3
BG
LINE 02 =
BG
* CARD 2
BG
LINE 04 =
BG
EOJ XYZ
BG
EXAMPLE SUCCESSFUL
BG
1I00A READY FOR COMMUNICATIONS.
BG
```

Figure 11. Console Listing of the SPOOL Macro Example

```
EXEC EXAMPLE
EXAMPLE BEGIN
EXAMPLE PUTSPOOL:
EXAMPLE GETSPOOL SEQUENTIAL:                          DATE . . . . . . . . .

// JOB XYZ
* CARD 2
* CARD 3
EOJ XYZ                                               DATE . . . . . . . . .

EXAMPLE GETSPOOL RANDOM RETRIEVAL:
LINE 03 =
* CARD 3
LINE 02 =
* CARD 2
LINE 04 =
EOJ XYZ
EXAMPLE SUCCESSFUL                                    DATE . . . . . . . . .
```

Figure 12. List Output of SPOOL Macro Example

Appendix B. Output Segmentation by SEGMENT Macro

The SEGMENT macro can be used for controlling output segmentation for a job running in a VSE/POWER-controlled partition. You can use the macro for the specification of new output characteristics that are to apply to the next segment. VSE/POWER assigns a new job number for the second and every subsequent SEGMENT request of your program.

The SEGMENT macro call results in a spooled I/O request for device=DEVADDR, and the macro call completion may in extreme cases be affected by storage or spool-space shortage of the VSE/POWER partition. The call opens a single-threaded path to VSE/POWER that ties up the Logical Transient Area for the duration of the simulated I/O request. Therefore, it is recommended to use the multi-threaded IPWSEGM macro instead, and even modify existing application programs to exchange the SEGMENT macro by the IPWSEGM call. Follow up the necessary conversion steps by comparing the SEGMENT coding examples 1 and 2 with the functionally equivalent examples 1 and 2 of the IPWSEGM macro.

When converting existing SEGMENT macro calls to IPWSEGM, then

- Use the FNO= and JNM= operands of the * $$ LST/PUN statements to achieve the function of the FORMS= and NAME= operands of the SEGMENT macro.
- No * $$ JOB statement can be passed via IPWSEGM to modify the **current** segment's name. This seldom used function is provided by the SEGMENT macro only.

SEGMENT Macro - Controlling Output Segmentation

Before using the macro, save the registers 0 and 1, because they are used and overwritten by VSE/POWER. Register 15 contains the return code passed by VSE/POWER on completion of the segment request.

Requirements for the Caller

AMODE:
24 or 31

RMODE:
24

ASC Mode:
Primary

Format of the Segment Macro

```
>>-+------+--SEGMENT DEVADDR=SYSxxx--+-------------------+----------->
   '-name-'                          '-,FORMS=formnumber-'

>--+---------------+--+---------------+-----------------------------><
   '-,JECL=-+-addr--+-' '-,NAME=-+-name--+-'
            '-(reg)-'            '-(reg)-'
```

DEVADDR=SYSxxx

For SYSxxx, specify the system or programmer logical unit assigned to the device on which the segmentation is to occur.

Your device specification in this operand must match your specification in the

LST operand of * $$ LST for list output, or

PUN operand of * $$ PUN for punch output

if you supplied a device specification in that statement. In case of mismatch, the LST/PUN operand values are ignored (see also Note 2 of the JECL operand).

FORMS=formnumber

For formnumber, specify the new one- to four-character form number which VSE/POWER is to use for the next segment.

VSE/POWER sets the form number to blanks if you omit the operand.

If you also use the JECL operand, any FORMS specification will be overwritten. In that case, use the FNO operand of your * $$ LST/PUN statement to define a form number for the next output segment.

JECL=address|(reg)

This operand points to a 71-byte area which contains one of the following JECL statements: * $$ LST, * $$ PUN, or * $$ JOB (see note 1). These statements are described in the *VSE/POWER Administration and Operation*, SC34-2625. The JECL area must reside below the 16MB line either in the partition or in the dynamic GETVIS area.

For `address`, specify the area's label in your program.

For `reg`, if you choose register notation, specify the register which contains the address of the area.

If you omit the specification of an * $$ LST, * $$ PUN, or * $$ JOB statement, default spooling values will be set for the new segment (regardless of any previously established values); therefore, pass only the operands needed to change default values which do not meet your requirements.

Note:

1. The SEGMENT macro causes new values to be set for the **new** (also called **next**) segment. However, passing a * $$ JOB statement with the JNM operand causes the **currently** processed segment to be renamed. The statement should, therefore, be passed by a separate SEGMENT macro after a previous SEGMENT request has created a new output segment with default or specified options. You may also specify the UINF operand within your * $$ JOB statement to provide new user information for the current segment.
2. Specification of the LST or PUN operand passed in * $$ LST or * $$ PUN statement is ignored.
3. If output segmentation is requested for output on an IBM 3800 printer, VSE/POWER uses the default printer setup for the new segment. If this is not desirable, supply an * $$ LST statement defining the desired printer setup. After having issued the SEGMENT macro, you may, in your program, issue a SETPRT macro requesting the proper printer setup.
4. Submitted * $$ LST or * $$ PUN statements with the operands TLBL= or LTAPE= cannot be used to specify spooling output for a VSE/SAM supported spool tape because this might result in a system softwait.

NAME=name|(reg)

The description of the NAME operand is provided for compatibility with releases previous to VSE/POWER 5.2. Instead, use the JNM operand of the * $$ LST/PUN statement, which provides the same function.

To assign a new name to the **new** (**next**) segment, place the JNM operand into the * $$ LST/PUN statement. To rename the **currently** processed segment, use the JNM operand of the * $$ JOB statement. Finally, pass the corresponding statement via the JECL= operand to the SEGMENT macro.

For name, specify a one- to eight-character new segment name. If you omit this operand, VSE/POWER uses the name by which the job was placed into the input queue. If you use register notation, the specified register must point to an eight-byte field containing the name of the segment. This field must reside below the 16MB line either in the partition or in the dynamic GETVIS area.

Return Codes from the SEGMENT Macro

Successful completion of the SEGMENT macro is indicated to the issuing program by a return code of 0 in register 15. If the operation fails, register 15 contains one of the return codes listed below.

Code **Meaning**

X'04' One of the following:

- The device specified in the DEVADDR operand is not spooled by VSE/POWER.
- VSE/POWER is not active.
- The partition in which your program is running is not under control of VSE/POWER.
- The spooled device is used for output with a disposition of N.
- The passed JECL statement was not one of:

  * $$ JOB

  * $$ LST

  * $$ PUN
- The passed JECL statement was incorrect and flagged on the console by message 1R33D, and the operator responded with FLUSH or EOB, or the corresponding automated action was triggered by the

  ```
  SET 1R33D=FLUSH|IGNORE
  ```

  statement.

X'08' VSE/POWER cannot accept the JECL statement because either:

- The partition was not started as a multitasking partition and the partition is waiting for work, or
- The partition was started as a multitasking partition and is waiting for work, but no JECL statement was submitted for the specified device.

Note:

1. When output segmentation is requested by the SEGMENT macro, all the already collected output by VSE/POWER for the specified device is added as an entry to the corresponding VSE/POWER queue - provided that the device has already been addressed during the VSE/POWER job before.
2. The output which has been created between two segment macros in a job transaction for the specified logical unit is added to a VSE/POWER

queue before the program reaches end-of-job. For long running programs like CICS, you can use the SEGMENT macro in a transaction to close spooling of output whenever desired. But, the specified output logical unit is unique in a CICS partition and, therefore, you may get mixed output if the same transaction runs twice at the same time, unless you have established private resource locking.

3. COBOL/VSE programs (and most likely all other LE/VSE languages) spool "double buffered" for unit record output, e.g. SYSLST. This causes problems if the VSE/POWER SEGMENT macro is used. The last line of the current segment may appear as the first line of the next segment instead. The two I/O buffers are handled by LIOCS (Logical Input/Output Control System) and are not synchronized with the SEGMENT macro call which expands into a SVC 0 and uses PIOCS (Physical Input/Output Control System).

   The solution to this problem is to select only one I/O buffer in the file definition of the calling high level language program in order to spool the data "single buffered".

Examples of the SEGMENT Macro

Example 1

The following example shows how to code the SEGMENT macro and its referenced data areas.

```
          ... ... ...
         LA    2,LSTCARD
         SEGMENT DEVADDR=SYSLST,JECL=(2)
          ... ... ...
LSTCARD  DC    CL71'* $$ LST  JNM=TESTOUT,FNO=ACB1,DISP=H'
          ... ... ...
```

Example 2

This sample job creates another VSE/POWER job in the RDR queue with new name and new input class using the DISP=I facility. To accomplish this, two SEGMENT macros are required.

The first SEGMENT macro passes a '* $$ PUN' JECL statement to VSE/POWER. This statement contains 'DISP=I' to indicate that the output segment being created should be added to the RDR queue. It also contains 'CLASS=7' and 'JNM=NEWJOB2' to specify the execution class of the new job segment with the unique name 'NEWJOB2'. When requesting this SEGMENT macro, VSE/POWER will create a new queue entry with the new class and jobname. The next step is to punch the job control and user data to the punch device. In this case, using physical I/O control (PIOCS). A second SEGMENT macro is required to have the current segment added to the RDR queue.

```
// JOB SEGMENT
// OPTION CATAL
// LIBDEF *,SEARCH=PRD1.MACLIB
// LIBDEF PHASE,CATALOG=IJSYSRS.SYSLIB
 PHASE SEGTEST,*
// EXEC ASSEMBLY,SIZE=100K
         CSECT
         PRINT GEN
         BALR  10,0
         USING *,10
* ------------------------------------------------------------------
* POINT TO THE * $$ PUN STATEMENT WITH DISP=I, THE NEW JOB CLASS AND
* THE NEW JOB NAME
* ------------------------------------------------------------------
```

```
         LA    2,PUNCARD
* -------------------------------------------------------------------
* NOW PASS 1ST SEGMENT REQUEST TO POWER WITH       >>> SEE NOTE 1 <<<
* -------------------------------------------------------------------
*
         SEGMENT DEVADDR=SYS008,JECL=(2)
*
         LTR   15,15                    DID IT WORK?
         BNZ   ERROR                    NO, GO INFORM OPERATOR
* -------------------------------------------------------------------
* NOW PASS THE JCL AND USER STATEMENTS FOR THE NEW POWER JOB
* BEING CREATED.
* -------------------------------------------------------------------
         LA    1,CCB                    POINT TO THE CCB
         EXCP  (1)                      AND ISSUE THE SVC0
         WAIT  (1)                      AND WAIT FOR I/O TO COMPLETE
* -------------------------------------------------------------------
* AND FINALLY, PASS SECOND SEGMENT REQUEST FOR THE PUNCH DEVICE
* TO HAVE THE NEW JOB ADDED TO THE READER QUEUE WITH UNIQUE
* EXECUTION CLASS AND JOB-NAME. AT THE SAME TIME RE-ESTABLISH DEFAULT
* PUNCH OUTPUT CHARACTERISTICS USING NO EXPLICIT * $$ PUN  JECL STMT.
*                                              >>> SEE NOTE 2 <<<
* -------------------------------------------------------------------
*
         SEGMENT DEVADDR=SYS008
*
         LTR   15,15                    DID IT WORK?
         BZ    EOJ                      YES, DONE, GOTO EOJ
ERROR    DS    0H
         C     15,FOUR                  IF RC ¬= 4 THEN
         BNE   CHK8                     CHECK FOR RC8
         MVI   RC,C'4'                  ELSE, SET RC4
         LA    1,CCB2                   AND INFORM THE OPERATOR
         EXCP  (1)                      VIA A CONSOLE MESSAGE
         WAIT  (1)                      WAIT FOR THE I/O TO COMPLETE
         B     EOJ                      AND GOTO EOJ
CHK8     DS    0H
         C     15,EIGHT                 IF RC ¬= 8 THEN
         BNE   UNKNOWN                  WE GOT AN UNKNOWN RETURN
         MVI   RC,C'8'                  ELSE, SET RC8
         LA    1,CCB2                   AND INFORM THE OPERATOR
         EXCP  (1)                      VIA A CONSOLE MESSAGE
         WAIT  (1)                      WAIT FOR THE I/O TO COMPLETE
         B     EOJ                      AND GOTO EOJ
UNKNOWN  DS    0H
         MVI   RC,C'U'                  IF RC ¬=4 AND RC ¬=8 THEN
         LA    1,CCB2                   SET UNKNOWN RETURN CODE
         EXCP  (1)                      ON THE CONSOLE
         WAIT  (1)                      WAIT FOR I/O TO COMPLETE
EOJ      DS    0H
         EOJ                            RETURN TO JOB CONTROL
* -------------------------------------------------------------------
*    * $$ PUN CARD WITH CLASS=7, DISP=I, AND JNM=NEWJOB2
* -------------------------------------------------------------------
*
PUNCARD  DC    CL71'* $$ PUN CLASS=7,DISP=I,JNM=NEWJOB2'
*
* -------------------------------------------------------------------
* SEGMENT MACRO ERROR MESSAGE TEXT
* -------------------------------------------------------------------
MSG1     DC    CL29'SEGMENT MACRO RETURN CODE IS '
* -------------------------------------------------------------------
* SEGMENT MACRO RETURN CODE TEXT
* -------------------------------------------------------------
* -------------------------------------------------------------------
RC       DC    CL1' '
         DS    0D
```

```
FOUR     DC    F'4'
EIGHT    DC    F'8'
* ---------------------------------------------------------------------
* CCB AND CCW FOR CONSOLE I/O
* ---------------------------------------------------------------------
CCB2     CCB   SYSLOG,CCWADDR2
CCWADDR2 CCW   09,MSG1,X'20',X'001E'
* ---------------------------------------------------------------------
* CCB AND CCWS FOR PUNCHING JCL AND USER STATEMENTS
* ---------------------------------------------------------------------
CCB      CCB   SYS008,CCWADDR
CCWADDR  CCW   01,BUF02,X'60',X'0050'
         CCW   01,BUF03,X'60',X'0050'
         CCW   01,BUF04,X'60',X'0050'
         CCW   01,BUF05,X'60',X'0050'
         CCW   01,BUF06,X'60',X'0050'
         CCW   01,BUF07,X'60',X'0050'
         CCW   01,BUF08,X'20',X'0050'
* ---------------------------------------------------------------------
* CONSTANTS FOR JOBSTREAM BEING PUNCHED
* ---------------------------------------------------------------------
BUF02    DC    CL80'// JOB NEWJOB2'
BUF03    DC    CL80'// PAUSE'
BUF04    DC    CL80'// EXEC LIBR'
BUF05    DC    CL80'A S=IJSYSRS.SYSLIB'
BUF06    DC    CL80'LD IPW$$NU.PHASE'
BUF07    DC    CL80'/*'
BUF08    DC    CL80'/&'
         END
/*
// EXEC LNKEDT
// ASSGN SYS008,SYSPCH
// LIBDEF PHASE,SEARCH=IJSYSRS.SYSLIB
// EXEC SEGTEST
/&
```

Example 3

The following example shows an ASSEMBLER subroutine which is called by a high level language (COBOL) program to invoke segmentation of its list output at a desired point in time.

```
 *
 *  . This routine is to be cataloged as .OBJ
 *
 *  . It will be AUTOLINKED to the CALLING program in the LNKEDT step
 *
 *  . It receives three parameters:
 *
 *     1) A 71-byte area containing the LST card with options for the
 *        new segment
 *     2) An 8-byte area containing the name for the new segment
 *     3) A full-word parameter where the return code from
 *        the SEGMENT macro (register 15 contents) will be placed
 *
 *  In the COBOL example below, the ASSEMBLER routine is used to
 *  segment output for different remote numbers (for instance,
 *  branch offices).
 *
 *
 *  Example of a COBOL calling program:
 *  -----------------------------------
 *
 *  WORKING-STORAGE SECTION.
 *      ...
 *  01  LST-CARD.
 *      05  FILLER     PIC  X(24)      VALUE '* $$ LST CLASS=T,REMOTE='.
 *      05  LST-REMID  PIC  9(03)      VALUE 0.
```

```
*      05  FILLER     PIC  X(44)      VALUE SPACES.
*  01  LST-NAME       PIC  X(08)      VALUE SPACES.
*  01  LST-RETCODE    PIC S9(04) COMP VALUE +0.
*      ...
*  PROCEDURE DIVISION.
*      ...
*      MOVE BRANCH-NUMBER TO LST-REMID.
*      MOVE BRANCH-NAME   TO LST-NAME.
*      CALL 'SEGMT' USING LST-CARD LST-NAME LST-RETCODE.
*      IF  LST-RETCODE NOT EQUAL ZERO
*          GO TO SEGMT-ERROR.
*      ...
*
*   called ASSEMBLER subroutine:
*----------------------------------------------------------------------
SEGMT    CSECT
         USING *,R2
         STM   R14,R12,12(R13)
         LR    R2,R15
         LM    R3,R5,0(R1)
*
         SEGMENT DEVADDR=SYSLST,JECL=(R3),NAME=(R4)
         ST    R15,0(R5)
         LM    R14,R12,12(R13)
         BR    R14
*
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
*
         END   SEGMT
```

Appendix C. Spool-Access Support Graphical Description

The following is a graphical summary of the Spool-Access Support similar to Backus Naur Format. It describes the Support communication protocol at various levels of abstraction using an informal language. The elements of the language are "words" and the description shows the "syntax" of the language.

A dictionary of the language occurs at the end of this section. An explanation of the description is given in the syntax diagram in Figure 13.

The description begins with a "General Phrase" which is a single "sentence" of the language describing the protocol at the highest level of abstraction. This is described in the syntax diagram in Figure 14.

The remaining detail of the "General Phase" follows below, which is in turn followed by the Dictionary.

Disclaimer

This description is meant to aid the user in understanding the protocol at various levels of abstraction. It is **not** meant to replace the textual description contained earlier in this publication. In case of a contradiction, the textual description should be referred to. IBM does not accept any responsibility for the accuracy of the protocol description.

Figure 13. Spool-Access Support Graphical Description Explanation

Layout of Graphical Description

```
                   (1)                          (2)
►►──User_Request──────┤ FRAGMENT ├──<VSE_Response>──────────────────────────────►◄
```

FRAGMENT:

```
├── . . . (protocol description detail) . . .──────────────────────────────────────┤
```

Notes:

1 User Request (Event)

2 VSE/POWER or VSE Response (State)

Figure 14. Spool-Access Support Graphical Description "General Phrase"

Spool-Access Support Description "General Phrase" Overview

Spool-Access Support Graphical Description

```
                                    ┌──────────────────────┐
                                    ▼                 (1)  │
►►──Identify──┬──Connect──┬──┤ USER_DIALOG ├──────────────┴──┬─Disconnect──────────┬──┬──┬──►◄
              │           │                                  ├─<PWR_Disconnect>────┤  │  │
              │           │                                  ├─<Severe_Err_AF>─────┤  │  │
              │           │                                  └─<Severe_Err_PWR>────┘  │  │
              │           │                   (2)                                     │  │
              │           ├─<PWR_Quiesced>────────────────────────────────────────────┤  │
              │           └─<Connect_Error>───────────────────────────────────────────┘  │
              └─<Identify_Error>─────────────────────────────────────────────────────────┘
```

Notes:

1 The Fragment or User event can be exited by any of the events or states following this location.

2 <PWR_Quiesced> occurs due to the event of VSE/POWER shutting down.

Spool-Access Support Description Detail

```
►►────────────────────────────────────────────────────────────────────►◄
```

USER_DIALOG:

```
├──┬──┤ PUT_SERVICE ├──┬──────────────────────────────────────────────┤
   ├──┤ GET_SERVICE ├──┤
   ├──┤ CTL_SERVICE ├──┤
   └──┤ GCM_SERVICE ├──┘
```

PUT_SERVICE:

```
├──┤ PUT_PROTOCOL ├──┬─────────────────────────────────────────┬──────┤
                     └─<Msg_Available>──┤ PUT_MSG_PROTOCOL ├──┘
```

PUT_PROTOCOL (Job or Output):

```
├──┤ PUT_OPEN ├──┬──┤ PUT_REQUEST ├──┬───────────────────┬──┬─────────┤
                 │                   └─<Msg_Available>───┘  │
                 ├─<Open_Err>───────────┬───────────────────┘
                 ├─<Severe_Err_AF>──────┤
                 └─<Severe_Err_PWR>─────┘
```

PUT_MSG_PROTOCOL:

```
   ┌─────────────────────────────────────────┐
   ▼                                    (1)  │                   (2)
├──Return_Msg──┬─────────────────────┬───────┴──┬─<Last_Msgs>──────┬──────┤
               ├─<Last_Msgs>─────────┤          ├─<End_of_Msgs>────┤
               ├─<End_of_Msgs>───────┤          ├─<More_Msgs>──────┤
               ├─<More_Msgs>─────────┤          ├─<Request_Err>────┤
               └─<Request_Err>───────┘          ├─<Severe_Err_AF>──┤
                                                └─<Severe_Err_PWR>─┘
```

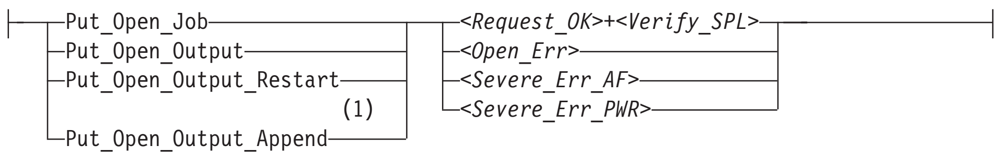
Notes:

1 The Fragment or User event can be exited by any of the events or states following this location.

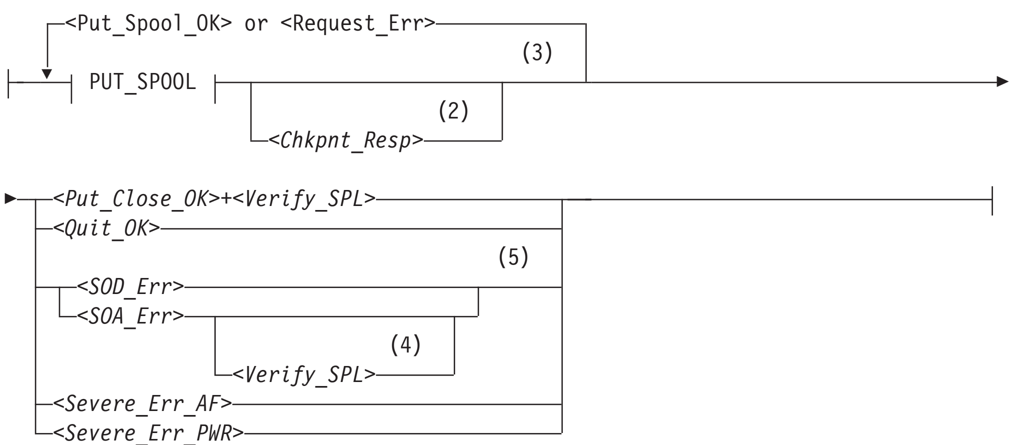
2 Any remaining messages will be discarded.

Spool-Access Support Description "General Phrase" Detail

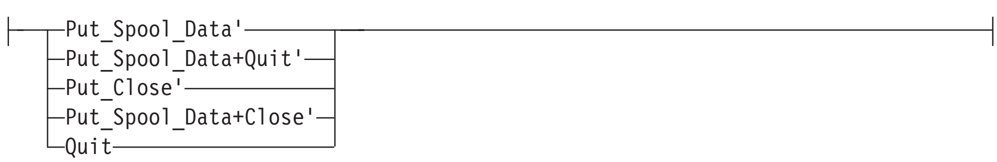
PUT_OPEN:



PUT_REQUEST:



PUT_SPOOL(Job):



PUT_SPOOL (Output):

```
├──┬─Put_Spool_Data──────────────────┬──────────────────────────────────────────────┤
   ├─Put_Checkpoint──────────────────┤
   ├─Put_Spool_Data+Checkpoint───────┤
   ├─Put_Restart─────────────────────┤
   ├─Put_Segmentation────────────────┤
   ├─Put_Spool_Data+Segmentation─────┤
   ├─Put_Segmentation+Update SPL─────┤
   ├─Put_Close───────────────────────┤
   ├─Put_Close_Append────────────────┤
   ├─Put_Close+Update SPL────────────┤
   ├─Put_Spool_Data+Quit─────────────┤
   ├─Put_Spool_Data+Close────────────┤
   ├─Put_Spool_Data+Close_Append─────┤
   ├─Get_OPTB────────────────────────┤
   ├─Modify_OPTB─────────────────────┤
   └─Quit────────────────────────────┘
```

Notes:

1 Output must have been closed by Put_Close_Append first.

2 Result of processing checkpoint request, or a restart request which was lower or equal to the last checkpoint.

3 The Fragment or User event can be exited by any of the events or states following this location.

4 If an SOA error occurs and the Put_Spool request contained more than one VSE/POWER job, then a Verification SPL is returned to the User to identify the job which failed to be submitted.

5 The SOD / SOA Error is indicated only if required as specified at request open with PWRSPL OPT=NOWAIT

Spool-Access Support Description "General Phrase" Detail

```
►►──────────────────────────────────────────────────────────────────────────►◄
```

GET_SERVICE:

```
├──┬─┤ GET_PROTOCOL ├────┬──────────────────────────────────────────────────┤
   └─┤ BROWSE_PROTOCOL ├─┘
```

GET_PROTOCOL(Job or Output):

```
├──┤ GET_OPEN ├──┬─<Open_OK>──┤ GET_REQUEST ├─┬────────────────────────────┤
                 ├─<Open_Err>─────────────────┤
                 ├─<Severe_Err_AF>────────────┤
                 └─<Severe_Err_PWR>───────────┘
```

GET_OPEN:

```
├──Get_Open───┬─<Open_OK>+<Verify_SPL>─┬──────────────────────────────────────────┤
              ├─<Open_Err>─────────────┤
              │            (1)         │
              ├─<SOA_Err>──────────────┤
              ├─<Severe_Err_AF>────────┤
              └─<Severe_Err_PWR>───────┘
```

GET_REQUEST:

```
    ┌─────────────────────────────────────────────────┐
    ▼                                                 │ (3)
├───┤ GET_SPOOL ├──┬─<Request_OK>─────────────────────┴─────┬─<Quit_OK>──────────┬──────┤
                   ├─<End_Of_Data>──────────────────┤        ├─<Get_Quit&Lock_OK> ┤
                   │                          (2)   │        ├─<Get_Close_OK>─────┤
                   ├─<Request_OK>+<Chkpnt_Resp>─────┤        ├─<Get_Purge_OK>─────┤
                   ├─<Request_OK>+<DSHR_SPL>────────┤        ├─<Severe_Err_AF>────┤
                   └─<Request_Err>──────────────────┘        └─<Severe_Err_PWR>───┘
```

GET_SPOOL:

```
├───┬─Get_Spool_Data────────┬──────────────────────────────────────────┤
    ├─Get_Checkpointing─────┤
    ├─Get_Checkpoint_Ext────┤
    ├─Get_Restart───────────┤
    ├─Get_Close─────────────┤
    ├─Get_Quit&Lock     ────┤
    ├─Get_Purge_Queue───────┤
    ├─Get_OPTB──────────────┤
    ├─Modify_OPTB───────────┤
    └─Quit──────────────────┘
```

Notes:

1 The SOD or SOA Error is indicated only if desired by the user as specified at request open with PWRSPL OPT=NOWAIT

2 Result of processing successful <Get_Checkpointing> or <Get_Checkpoint_Ext> request.

3 The Fragment or User event can be exited by any of the events or states following this location.

Spool-Access Support Description "General Phrase" Detail

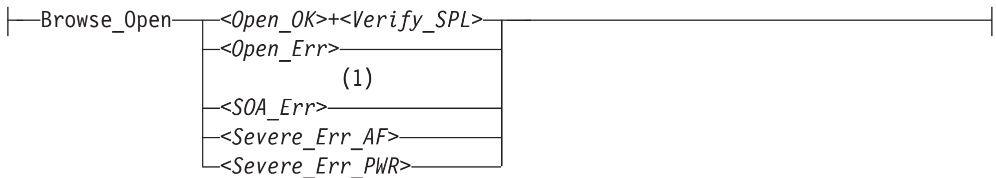
```
►►──────────────────────────────────────────────────────────────────►◄
```

BROWSE_PROTOCOL(Job or Output):

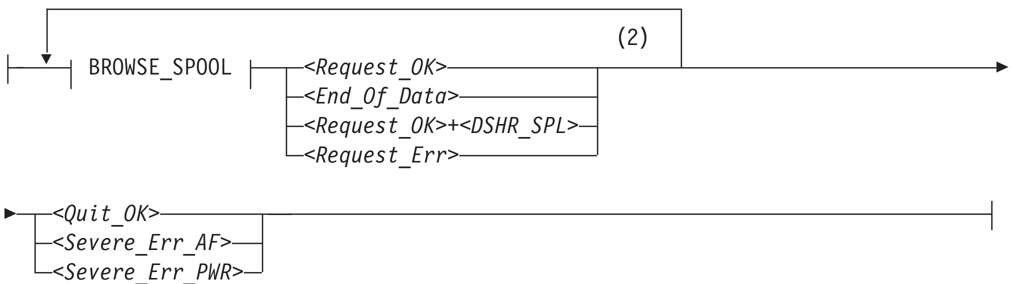
```
├──┤ BROWSE_OPEN ├──┬─<Open_OK>──┤ BROWSE_REQUEST ├─┬──────────────────┤
                    └─┬─<Open_Err>────────────┬─────┘
                      ├─<Severe_Err_AF>───────┤
                      └─<Severe_Err_PWR>──────┘
```

Spool-Access Support Graphical Description

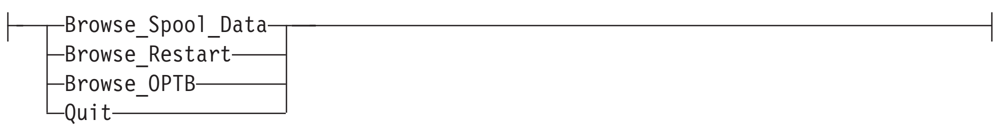
BROWSE_OPEN:



BROWSE_REQUEST:



BROWSE_SPOOL:

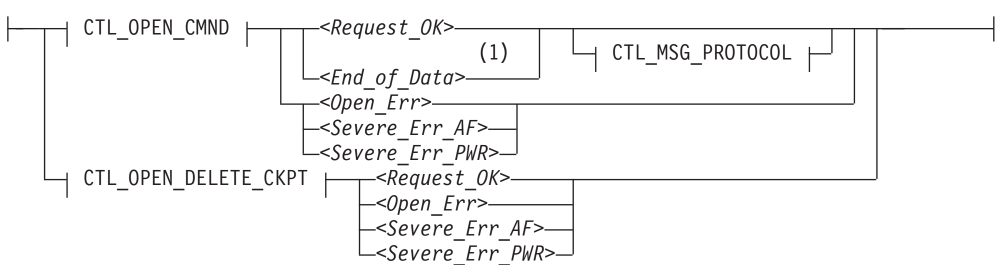


Notes:

1. The SOD or SOA Error is indicated only if desired by the user as specified at request open with PWRSPL OPT=NOWAIT
2. The Fragment or User event can be exited by any of the events or states following this location.

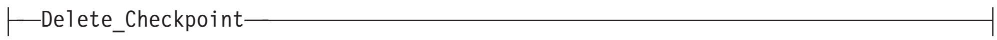
Spool-Access Support Description "General Phrase" Detail

CTL_SERVICE:

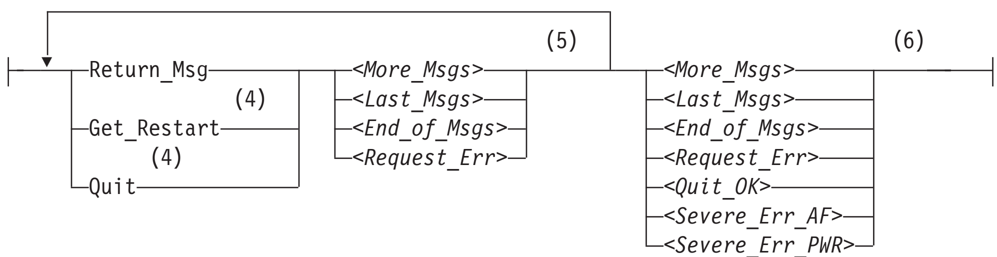


CTL_OPEN_CMND:

CTL_OPEN_DELETE_CKPT:



CTL_MSG_PROTOCOL:



Notes:

1 <End_of_Data> indicated instead of message "1R88I OK".

2 These commands can be specified using the PWRSPL FUNC= operand.

3 This represents all commands that can be specified using PWRSPL FUNC=COMMAND.

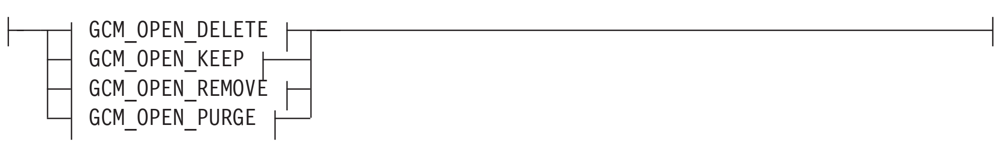
4 The Get_Restart and Quit requests may be issued during processing of command response message buffer(s) for a successful "PDISPLAY queue" command.

5 The Fragment can be exited by any of the following states.

6 Any remaining messages will be discarded if the <End_of_Data> state has not been indicated.

Spool-Access Support Description "General Phrase" Detail

GCM_SERVICE:



Spool-Access Support Graphical Description

GCM_OPEN_DELETE:

```
                                                 (1)
├──GCM_Open_Delete──┬─<Request_OK>──┤ GCM_DELETE_REQ ├──┬──────────────────┬─┬──────┤
                    │                                   └─<Severe_Err_AF>──┤ │
                    │                                   └─<Severe_Err_PWR>─┘ │
                    └─<Open_Err>─────────────────────────────────────────────┘
                    ├─<Severe_Err_AF>──┤
                    └─<Severe_Err_PWR>─┘
```

GCM_OPEN_KEEP:

```
                                                 (1)
├──GCM_Open_Keep──┬─<Request_OK>──┤ GCM_KEEP_REQ ├──┬──────────────────┬─┬──────┤
                  │                                 ├─<Severe_Err_AF>──┤ │
                  │                                 └─<Severe_Err_PWR>─┘ │
                  └─<Open_Err>───────────────────────────────────────────┘
                  ├─<Severe_Err_AF>──┤
                  └─<Severe_Err_PWR>─┘
```

GCM_OPEN_REMOVE:

```
├──GCM_Open_Remove──┬─<Request_OK>─────┬──────────────────────────┤
                    ├─<Open_Err>───────┤
                    ├─<Severe_Err_AF>──┤
                    └─<Severe_Err_PWR>─┘
```

GCM_OPEN_PURGE:

```
├──GCM_Open_Purge──┬─<Request_OK>─────┬──────────────────────────┤
                   ├─<Open_Err>───────┤
                   ├─<Severe_Err_AF>──┤
                   └─<Severe_Err_PWR>─┘
```

Notes:

1 The Fragment or User event can be exited by any of the events or states following this location. The GCM Service does not require a specific exit state (e.g. Quit or Close) as other Services.

Spool-Access Support Description "General Phrase" Detail

```
►►──────────────────────────────────────────────────────────────►◄
```

GCM_DELETE_REQ:

```
     ┌──────────────────────────────────┐
     ▼                         (1)      │
├────GCM_More──┬─<More_Msgs>────────────┴──┬──────────────────┬──────┤
               ├─<Last_Msgs>────┤          ├─<Severe_Err_AF>──┤
               ├─<End_of_Msgs>──┤          └─<Severe_Err_PWR>─┘
               └─<Request_Err>──┘
```

GCM_KEEP_REQ:

```
   ┌────────────────────────────────────────────┐
   ▼                                        (1) │
├────┬─GCM_Remove──┬─<End_of_Data>──┬──┬────────┴──┬──────────────────────┬──┤
     │             └─<Request_Err>──┘  │           └─┬─<Severe_Err_AF>──┬─┘
     │   ┌─────────────────────────┐   │             └─<Severe_Err_PWR>─┘
     │   ▼                         │   │
     └─────GCM_More──┬─<More_Msgs>─┴───┘
                     ├─<Last_Msgs>───┤
                     ├─<End_of_Msgs>─┤
                     └─<Request_Err>─┘
```

Notes:

1 The Fragment or User event can be exited by any of the events or states following this location. The GCM Service does not require a specific exit state (e.g. Quit or Close) as other Services.

Spool-Access Support Description "Dictionary"

User Requests (Miscellaneous)
Definition

1.1.Identify

User performs XPCC FUNC=IDENT function.
(for details, see "Set Up a Communication Path")

1.2.Connect

User performs XPCC FUNC=CONNECT function.
(for details, see "Set Up a Communication Path")

1.3.Disconnect

User performs XPCC FUNC=DISCONN or FUNC=DISCPRG function.
(for details, see "Ending Access to VSE/POWER Services")

1.4.Return_Msg

User requests VSE/POWER messages from previous PUT_PROTOCOL or CTL_PROTOCOL request.

`PXUACT1=PXUATRMR PXUBTYP=0 IJBXBLN=0`

1.5.Quit

User issues a Quit request.

`PXUACT1=PXUATABR PXUBTYP=0 IJBXBLN=0`

User Requests (Put Service Job)

2.1.Put_Open_Job'

User PUT-OPEN Job request.

PXUACT1=0	PXUBTYP=SPL	SPLGRQB=SPLGRPUT SPLGQI=SPLGQIR

2.2.Put_Spool_Data'
User PUT-SPOOL Job data request.

```
PXUACT1=0          PXUBTYP=PXUBTNBL
```

2.3.Put_Spool_Data+Close'
User PUT-SPOOL Job data and a PUT-CLOSE Job request

```
PXUACT1=PXUATEOD  PXUBTYP=PXUBTNDB
```

2.4.Put_Spool_Data+Quit'
User PUT-SPOOL Job data and a PUT-QUIT Job request

```
PXUACT1=PXUATABR  PXUBTYP=PXUBTNDB
```

2.5.Put_Close'
User PUT-CLOSE Job Service request.

```
PXUACT1=PXUATEOD  PXUBTYP=0  IJBXBLN=0
```

User Requests (Put Service Output)

3.1.Put_Checkpoint
User PUT-CHECKPOINT Output request.

```
PXUACT1=PXUATCHK  PXUBTYP=0
```

3.2.Put_Close
User PUT-CLOSE Output Service request.

```
IJBXBLN=0
PXUACT1=PXUATEOD  PXUBTYP=0
```

3.3.Put_Close_Append

User PUT-CLOSE Output request (for later PUT-OPEN append).

```
PXUACT1=PXUATROE  PXUBTYP=0
```

3.4.Put_Close+Update SPL

User PUT-CLOSE Output request (for later PUT-OPEN append) with update SPL.

```
PXUACT1=PXUATROE  PXUBTYP=PXUBTSPL
```

3.5.Put_Open_Output
User PUT-OPEN Output request.

```
PXUACT1=0          PXUBTYP=PXUBTSPL   SPLGRQB=SPLGRPUT
                                      SPLGQI=SPLGQIL/P
```

3.6.Put_Open_Output_Append
User PUT-OPEN request for appending Output data.

```
PXUACT1=0          PXUBTYP=PXUBTSPL       etc.
```

3.7.Put_Open_Output_Restart
User PUT-OPEN request for restarting Output data.

```
PXUACT1=0          PXUBTYP=PXUBTSPL       etc.
```

3.8.Put_Restart
User PUT-RESTART Output Service request.

```
PXUACT1=0          PXUBTYP=PXUBTCTL
```

3.9.Put_Segmentation
User PUT-SEGMENTATION Output Service request.

```
PXUACT1=PXUATSGM  PXUBTYP=0     IJBXBLN=0
```

3.10.Put_Segmentation+Update_SPL

User PUT-SEGMENTATION Output Service request with update SPL

`PXUACT1=PXUATSGM  PXUBTYP=PXUBTSPL`

3.11.Put_Spool_Data

User PUT-SPOOL Output data request.

`PXUACT1=PXUATSGM  PXUBTYP=PXUBTNDB`

3.12.Put_Spool_Data+Close

User PUT-SPOOL data and PUT-CLOSE Output request.

`PXUACT1=PXUATEOD  PXUBTYP=PXUBTNDB`

3.13.Put_Spool_Data+Checkpoint

User PUT-SPOOL data and PUT-CHECKPOINT Output request.

`PXUACT1=PXUATCHK  PXUBTYP=PXUBTNDB`

3.14.Put_Spool_Data+Quit

User PUT-SPOOL data and PUT-QUIT Output request.

`PXUACT1=PXUATABR  PXUBTYP=PXUBTNDB`

3.15.Put_Spool_Data+Segmentation

User PUT-SPOOL data and PUT-SEGMENTATION Output request.

`PXUACT1=PXUATSGM  PXUBTYP=PXUBTNDB`

3.16.Put_Spool_Data+Close_Append

User PUT-SPOOL data and PUT-CLOSE Output request (for later PUT-OPEN append).

`PXUACT1=PXUATROE  PXUBTYP=PXUBTNDB`

User Requests (Get Service, Job or Output) Definition

4.1.Get_Open

User GET-OPEN Service request.

```
PXUACT1=0  PXUBTYP=PXUBTSPL             SPLGRQB=SPLGRGET
                                        SPLGQI=SPLGQIR/L/P
```

Note: For Direct Queue Entry Access specify additionally

`SPLXQNUM=queue-entry-number        SPLGOPT2.SPLGO2QN=ON`

4.2.Get_Spool_Data

User GET-SPOOL Data Service request.

`PXUACT1=PXUATSDR  PXUBTYP=0         IJBXBLN=0`

4.3.Get_Close

User GET-CLOSE Service request.

`PXUACT1=PXUATRQS  PXUBTYP=0         IJBXBLN=0`

4.4.Quit&Lock

User GET-QUIT-and-LOCK Service request.

`PXUACT1=PXUAT1PF  PXUBTYP=0         IJBXBLN=0`

4.5.Get_Purge_Queue

User GET-PURGE Service request.

`PXUACT1=PXUATPRG  PXUBTYP=0         IJBXBLN=0`

4.6.Get_Checkpointing

User GET-CHECKPOINTING Service request.

```
PXUACT1=0        PXUBTYP=PXUBTCTL
Send Buffer=Checkpoint control record
```

4.7.Get_Checkpoint_Ext

User GET-CHECKPOINT_EXT Service request.
(Get Extended Checkpoint Information)

```
PXUACT1=PXUATCKR  PXUBTYP=0        IJBXBLN=0
```

4.8.Get_Restart

User GET-RESTART Service request.

```
PXUACT1=0        PXUBTYP=PXUBTCTL
```

4.9.Get_OPTB

User GET-OPTB Output request

```
PXUACT1=0        PXUBTYP=PXUBTCTL
Send Buffer=GET-OPTB control record
```

4.10.Modify_OPTB

User Modify-OPTB Output request

```
PXUACT1=0        PXUBTYP=PXUBTCTL
Send Buffer=Modify-OPTB control record
```

User Requests (Browse Service, Job or Output) Definition

5.1.Browse_Open

User BROWSE-OPEN Service request.

```
PXUACT1=0  PXUBTYP=PXUBTSPL            SPLGRQB=SPLGRGET
                                       SPLGFB1=SPLGF1BR
                                       SPLGQI=SPLGQIR/L/P
```

Note:

1. For direct queue entry access specify additionally

   ```
   SPLXQNUM=queue-entry-number          SPLGOPT2.SPLGO2QN=ON
   ```

2. For direct access to queue entries in creation, specify additionally

   ```
   SPLXQNUM=queue-entry-number          SPLGOPT2.SPLGO2QN=ON
                                        SPLGOPT.SPLGOGIC=ON
   ```

5.2.Browse_Spool_Data

User BROWSE-SPOOL Data Service request.

```
PXUACT1=PXUATSDR  PXUBTYP=0        IJBXBLN=0
```

5.3.Browse_Restart

User BROWSE-RESTART Service request.

```
PXUACT1=0        PXUBTYP=PXUBTCTL
```

5.4.Browse_OPTB

User BROWSE-OPTB Output request.

```
PXUACT1=0        PXUBTYP=PXUBTCTL
Send Buffer=Get-OPTB control record
```

User Requests (CTL Sevice) Definition

6.1.PALTER

User requests VSE/POWER to perform command 'PALTER' (PWRSPL FUNC=(ALTER,attrib-type),NEWVAL=field)

```
PXUACT1=0 PXUBTYP=SPL  SPLGSRB=SPLGSRAL SPLGRQB=SPLGRCTL
                       SPLGFB2=attrib-type SPLGNV=field
                                        SPLGQI=SPLGQIR/L/P/X
```

Note: For Direct Queue Entry Access specify additionally

```
SPLXQNUM=queue-entry-number       SPLGOPT2.SPLGO2QN=ON
```

6.2.PCANCEL

User requests VSE/POWER to perform command 'PCANCEL' (PWRSPL FUNC=CANCEL)

```
PXUACT1=0 PXUBTYP=SPL  SPLGSRB=SPLGSRCN SPLGRQB=SPLGRCTL
                       SPLGJB=jobname   SPLGJN=jobnumber
```

6.3.PDELETE

User requests VSE/POWER to perform command 'PDELETE' (PWRSPL FUNC=DELETE)

```
PXUACT1=0 PXUBTYP=SPL  SPLGSRB=SPLGSRCL SPLGRQB=SPLGRCTL
                       SPLGJB=jobname   SPLGJN=jobnumber
                                        SPLGQI=SPLGQIR/L/P/X
```

Note: For Direct Queue Entry Access specify additionally

```
SPLXQNUM=queue-entry-number       SPLGOPT2.SPLGO2QN=ON
```

6.4.PDISPLAY

User requests VSE/POWER to perform command 'PDISPLAY' (PWRSPL FUNC=DISPLAY)

```
PXUACT1=0 PXUBTYP=SPL  SPLGSRB=SPLGSRDY SPLGRQB=SPLGRCTL
                       SPLGJB=jobname   SPLGJN=jobnumber
                                        SPLGQI=SPLGQIR/L/P/X
```

Note: For Direct Queue Entry Access specify additionally

```
SPLXQNUM=queue-entry-number       SPLGOPT2.SPLGO2QN=ON
```

User Requests (CTL Sevice) Definition

6.5.PHOLD

User requests VSE/POWER to perform command 'PHOLD' (PWRSPL FUNC=HOLD)

```
PXUACT1=0 PXUBTYP=SPL  SPLGSRB=SPLGSRHD SPLGRQB=SPLGRCTL
                       SPLGJB=jobname   SPLGJN=jobnumber
                                        SPLGQI=SPLGQIR/L/P/X
```

Note: For Direct Queue Entry Access specify additionally

```
SPLXQNUM=queue-entry-number       SPLGOPT2.SPLGO2QN=ON
```

6.6.PRELEASE

User requests VSE/POWER to perform command 'PRELEASE' (PWRSPL FUNC=RELEASE)

```
PXUACT1=0 PXUBTYP=SPL  SPLGSRB=SPLGSRRL SPLGRQB=SPLGRCTL
                       SPLGJB=jobname   SPLGJN=jobnumber
                                        SPLGQI=SPLGQIR/L/P/X
```

Note: For Direct Queue Entry Access specify additionally

```
SPLXQNUM=queue-entry-number      SPLGOPT2.SPLGO2QN=ON
```

6.7.Command

User requests VSE/POWER to perform command as specified in the field SPLCFLD.

(PWRSPL FUNC=COMMAND)

```
PXUACT1=0 PXUBTYP=SPL  SPLGSRB=SPLGSRCM SPLGRQB=SPLGRCTL
                       SPLCFLD=(desired command)
```

Note: For Direct Queue Entry Access specify additionally

```
SPLXQNUM=queue-entry-number      SPLGOPT2.SPLGO2QN=ON
```

6.8.Delete_Checkpoint

User CTL request to delete a checkpoint.

```
PXUACT1=0 PXUBTYP=SPL  SPLGSRB=SPLGSRDC SPLGRQB=SPLGRCTL
                       SPLGJB=jobname   SPLGJN=jobnumber
SPLXQNUM=queue-entry-number             SPLGQI=SPLGQIR/L/P/X
```

User Requests (GCM Sevice)
Definition

7.1.GCM_Open_Delete

User GCM-OPEN-DELETE request.

```
PXUBTYP=PXUBTSPL SPLGRQB=SPLGRGCM SPLGFB1=SPLGF1DM
```

7.2.GCM_Open_Keep

User GCM-OPEN-KEEP request.

```
PXUBTYP=PXUBTSPL SPLGRQB=(etc.)   SPLGFB1=SPLGF1KM
```

7.3.GCM_Open_Remove

User GCM-OPEN-REMOVE request.

```
PXUBTYP=PXUBTSPL SPLGRQB=(etc.)   SPLGFB1=SPLGF1RM
```

7.3.GCM_Open_Purge

User GCM-OPEN-PURGE request.

```
PXUBTYP=PXUBTSPL SPLGRQB=(etc.)   SPLGFB1=SPLGF1PM
```

7.5.GCM_More

User GCM-MORE subrequest.

```
PXUBTYP=0        PXUACT1=PXUATGCM  IJBXBLN=0
```

7.6.GCM_Remove

User GCM-REMOVE subrequest.

```
PXUBTYP=0        PXUACT1=PXUATDEL  IJBXBLN=0
```

System Responses
Definition

8.1.<Connect_Error>

XPCC FUNC=CONNECT error.

8.5a.<Msg_Available>

VSE/POWER response that message(s) are available for the user to fetch following a PUT_PROTOCOL.

```
PXPIMSG=On
```

8.2.<Chkpnt_Resp>

Response of VSE/POWER to an accepted Checkpoint request.

`Reply Buffer: Checkpoint Response Record`

8.3.<DSHR_SPL>
Response of VSE/POWER to GET-SPOOL data request that returns a data-set-header record in the form of a SPL to the user's reply buffer.

`Reply Buffer: SPL`

8.4.<End_of_Data>
End of VSE/POWER response data (GET-SPOOL data or messages: PUT, CTL or GCM).

`PXPRETCD=X'00'  PXPFBKCD=X'01'`

8.5.<Identify_Error>
XPCC FUNC=IDENT error.

8.6a.<Msg_Available>
VSE/POWER response that message(s) are available for the User to fetch following a PUT_PROTOCOL.

`PXPIMSG=On`

8.6b.<More_Msgs>
VSE/POWER messages returned in buffer in response to a User request.

`PXPRETCD=X'00'  PXPFBKCD=X'00'    IJBXSLN > 0`

8.6c.<Last_Msgs>
Final VSE/POWER messages returned in buffer in response to a User request.

`PXPRETCD=X'00'  PXPFBKCD=X'01'    IJBXSLN > 0`

8.6c.<End_of_Msgs>
No VSE/POWER messages available in buffer in response to a User request.

`PXPRETCD=X'00'  PXPFBKCD=X'01'    IJBXSLN = 0`

8.7.<Open_Err>
VSE/POWER has detected a User OPEN request error.

`PXPRETCD=X'04',X'08' PXPFBKCD=(see Manual)`

8.7a.<Put_Close_Ok>
VSE/POWER has accepted a PUT-CLOSE user request.

`PXPRETCD=X'00'`

8.7b.<Put_Spool_OK>
VSE/POWER has accepted a PUT-SPOOL user request.

`PXPRETCD=X'00'`

8.7c.<Put_Spool_Err>
VSE/POWER has rejected a user PUT request (except for <SOD_Err> or <SOA_Err>).

`PXPRETCD=X'04',X'08'  (except for <SOD_Err> or <SOA_Err> )`

System Responses
Definition

8.8a.<PWR_Disconnect>
VSE/POWER has disconnected.

8.8b.<PWR_Quiesced>
VSE/POWER has quiesced.

8.9.<Quit_OK>
VSE/POWER has accepted a QUIT user request.

`PXPRETCD=X'00'`

8.10a.<Request_Err>
VSE/POWER has detected a User request error.

`PXPRETCD=X'04',X'08' (except for <SOD_Err> or <SOA_Err>)`

8.10b.<Request_OK>
VSE/POWER response to a User request indicating that no warning or error has occurred. When processing response message(s), this state indicates that (more) response messages are queued for the user to process.

`PXPRETCD=X'00'  PXPFBKCD=X'00'`

8.11a.<SOD_Err>
"Short-on-Data-File" error, i.e. VSE/POWER has run out of Data File storage.

`PXPRETCD=X'04'  PXP04SOD=X'08'`

8.11b.<SOA_Err>
"Short-on-Account-File" error, i.e. VSE/POWER has run out of Account File storage.

`PXPRETCD=X'04'  PXP04SOA=X'09'`

8.11c.<Severe_Err_PWR>
A severe error has been detected by VSE/POWER or VSE/POWER has stopped the communication, i.e.:

- A User Request error has occurred.
- A User Dialog request error has occurred.
- A VSE/POWER PSTOP SAS command has been issued.
- A VSE/POWER Queue/Data File disk I/O error has occurred.

`PXPRETCD   >X'08'`

8.11d.<Severe_Err_AF>
A severe User error has been detected by z/VSE following the XPCC call or VSE/POWER has terminated, causing the XPCC request to be rejected:

- immediately following the XPCC call.

  `Register 15 > 0`
- following the wait on the XPCC call completion when the user is posted.

  `IJBXREAS=error`

8.12.<Verify_SPL>
Response of VSE/POWER that returns an SPL to the user's reply buffer containing the up-to-date information about the indicated queue entry.

`Reply Buffer: SPL`

Glossary

This glossary includes terms and definitions for IBM z/VSE.

The following cross-references are used in this glossary:
1. See refers the reader from a term to a preferred synonym, or from an acronym or abbreviation to the defined full form.
2. See also refers the reader to a related or contrasting term.

To view glossaries for other IBM products, go to www.ibm.com/software/globalization/terminology.

A

Access Control Logging and Reporting. An IBM licensed program to log all attempts of access to protected data and to print selected formatted reports on such attempts.

access control table (DTSECTAB). A table that is used by the system to verify a user's right to access a certain resource.

access list. A table in which each entry specifies an address space or data space that a program can reference.

access method. A program, that is, a set of commands (macros) to define files or addresses and to move data to and from them; for example VSE/VSAM or VTAM.

account file. A disk file that is maintained by VSE/POWER containing accounting information that is generated by VSE/POWER and the programs running under VSE/POWER.

addressing mode (AMODE). A program attribute that refers to the address length that a program is prepared to handle on entry. Addresses can be either 24 bits or 31 bits in length. In 24 bit addressing mode, the processor treats all virtual addresses as 24-bit values; in 31 bit addressing mode, the processor treats all virtual addresses as 31-bit values. Programs with an addressing mode of ANY can receive control in either 24 bit or 31 bit addressing mode.

administration console. In z/VSE, one or more consoles that receive all system messages, except for those that are directed to one particular console. Contrast this with the user console, which receives only those messages that are directed to it, for example messages that are issued from a job that was submitted with the request to echo its messages to that console. The operator of an administration console can reply to all outstanding messages and enter all system commands.

alternate block. On an FBA disk, a block that is designated to contain data in place of a defective block.

alternate index. In systems with VSE/VSAM, the index entries of a given base cluster that is organized by an alternate key, that is, a key other than the prime key of the base cluster. For example, a personnel file preliminary ordered by names can be indexed also by department number.

alternate library. An interactively accessible library that can be accessed from a terminal when the user of that terminal issues a connect or switch library request.

alternate track. A library, which becomes accessible from a terminal when the user of that terminal issues a connect or switch (library) request.

AMODE. Addressing mode.

APA. All points addressable.

APAR. Authorized Program Analysis Report.

appendage routine. A piece of code that is physically located in a program or subsystem, but logically and extension of a supervisor routine.

application profile. A control block in which the system stores the characteristics of one or more application programs.

application program. A program that is written for or by a user that applies directly to the user's work, such as a program that does inventory control or payroll. See also batch program and online application program.

AR/GPR. Access register and general-purpose register pair.

ASC mode. Address space control mode.

ASI (automated system initialization) procedure. A set of control statements, which specifies values for an automatic system initialization.

attention routine (AR). A routine of the system that receives control when the operator presses the Attention key. The routine sets up the console for the input of a command, reads the command, and initiates the system service that is requested by the command.

automated system initialization (ASI). A function that allows control information for system startup to be cataloged for automatic retrieval during system startup.

autostart. A facility that starts VSE/POWER with little or no operator involvement.

auxiliary storage. Addressable storage that is not part of the processor, for example storage on a disk unit. Synonymous with external storage.

B

B-transient. A phase with a name beginning with $$B and running in the Logical Transient Area (LTA). Such a phase is activated by special supervisor calls.

bar. 2 GigyByte (GB) line

basic telecommunications access method (BTAM). An access method that permits read and write communication with remote devices. BTAM is not supported on z/VSE.

BIG-DASD. A subtype of Large DASD that has a capacity of more than 64 K tracks and uses up to 10017 cylinders of the disk.

block. Usually, a block consists of several records of a file that are transmitted as a unit. But if records are very large, a block can also be part of a record only. On an FBA disk, a block is a string of 512 bytes of data. See also a control block.

block group. In VSE/POWER, the basic organizational unit for fixed-block architecture (FBA) devices. Each block group consists of a number of 'units of transfer' or blocks.

C

CA splitting. Is the host part of the VSE JavaBeans, and is started using the job STARTVCS, which is placed in the reader queue during installation of z/VSE. Runs by default in dynamic class R. In VSE/VSAM, to double a control area dynamically and distribute its CIs evenly when the specified minimum of free space get used up by more data.

carriage control character. The fist character of an output record (line) that is to be printed; it determines how many lines should be skipped before the next line is printed.

catalog. A directory of files and libraries, with reference to their locations. A catalog may contain other information such as the types of devices in which the files are stored, passwords, blocking factors. To store a library member such as a phase, module, or book in a sublibrary. See also VSE/VSAM catalog.

cell pool. An area of virtual storage that is obtained by an application program and managed by the callable cell pool services. A cell pool is located in an address space or a data space and contains an anchor, at least one extent, and any number of cells of the same size.

central location. The place at which a computer system's control device, normally the systems console in the computer room, is installed.

chained sublibraries. A facility that allows sublibraries to be chained by specifying the sequence in which they must be searched for a certain library member.

chaining. A logical connection of sublibraries to be searched by the system for members of the same type (phases or object modules, for example).

channel command word (CWW). A doubleword at the location in main storage that is specified by the channel address word. One or more CCWs make up the channel program that directs data channel operations.

channel program. One or more channel command words that control a sequence of data channel operations. Execution of this sequence is initiated by a start subchannel instruction.

channel scheduler. The part of the supervisor that controls all input/output operations.

channel subsystem. A feature of 370-XA and Enterprise Systems Architecture that provides extensive additional channel (I/O) capabilities over the System/370.

channel to channel attachment (CTCA). A function that allows data to be exchanged

1. Under the control of VSE/POWER between two virtual VSE machines running under VM or
2. Under the control of VTAM between two processors.

character-coded request. A request that is encoded and transmitted as a character string. Contrast with *field-formatted request*.

checkpoint.

1. A point at which information about the status of a job and the system can be recorded so that the job step can be restarted later.
2. To record such information.

CICS (Customer Information Control System). An IBM program that controls online communication between terminal users and a database. Transactions that are entered at remote terminals are processed concurrently by user-written application programs. The program includes facilities for building, using, and servicing databases.

CICS ECI. The CICS External Call Interface (ECI) is one possible requester type of the *CICS business logic interface* that is provided by the CICS Transaction Server for VSE/ESA. It is part of the CICS client and allows workstation programs to CICS function on the z/VSE host.

CICS EXCI. The EXternal CICS Interface (EXCI) is one possible requester type of the *CICS business logic interface* that is provided by the CICS Transaction Server for VSE/ESA. It allows any BSE batch application to call CICS functions.

CICS system definition (CSD) file. Is the host part of the VSE JavaBeans, and is started using the job STARTVCS, which is placed in the reader queue during installation of z/VSE. Runs by default in dynamic class R. See CSD.

CICS Transaction Server for VSE/ESA. A z/VSE base program that controls online communication between terminal users and a database. This is the successor system to CICS/VSE.

CICS TS. CICS Transaction Server

CICS/VSE. Customer Information Control System/VSE. No longer shipped on the Extended Base Tape and no longer supported, cannot run on z/VSE 5.1.

class. In VSE/POWER, a group of jobs that either come from the same input device or go to the same output device.

cluster controller. A hardware unit to control the input/output operations of more than one device that is connected to it. A cluster controller might be run by a program that is stored and executed in the unit; for example, the IBM 3601 Finance Communication Controller. Or it might be controlled entirely by hardware; for example, the IBM 3272 Control Unit.

Common Connector Framework (CCF). Is part of IBM's *Visual Age for Java*, and allows connections to remote hosts to be created and maintained. The CCF classes are contained in the VSEConnector.jar file and are used internally by the VSE JavaBeans. CCF is important for multitier architectures where, for example, servlets run on a middle-tier platform. Because CCF allows open connections to be kept in a pool, this avoids the time that is involved in opening and closing TCP/IP connection to the remote z/VSE host each time a servlet is invoked.

CMS. Conversational monitor system running on z/VM.

common library. A library that can be interactively accessed by any user of the (sub)system that owns the library.

communication adapter. A circuit card with associated software that enables a processor, controller, or other device to be connected to a network.

communication region. An area of the supervisor that is set aside for transfer of information within and between programs.

component.

1. Hardware or software that is part of a computer system.
2. A functional part of a product, which is identified by a component identifier.
3. In z/VSE, a component program such as VSE/POWER or VTAM.
4. In VSE/VSAM, a named, cataloged group of stored records, such as the data component or index component of a key-sequenced file or alternate index.

component identifier. A 12-byte alphanumeric string, uniquely defining a component to MSHP.

conditional job control. The capability of the job control program to process or to skip one or more statements that are based on a condition that is tested by the program.

connect. To authorize library access on the lowest level. A modifier such as "read" or "write" is required for the specified use of a sublibrary.

connection pooling. Introduced with an z/VSE 5.1 update to manage (reuse) connections of the z/VSE database connector in CICS TS.

ConnectionManager class. Is part of CCF, and identifies the connection to a remote z/VSE host: it holds connections between the middle-tier and the remote z/VSE server. Servlets can reserve a connection from the pool, work with it and give it back later. This is performed internally using VSE JavaBeans.

connector. In the context of z/VSE, a connector provides the middleware to connect two platforms: Web Client and z/VSE host, middle-tier and z/VSE host, or Web Client and middle-tier.

connector (e-business connector). A piece of software that is provided to connect to heterogeneous environments. Most connectors communicate to non-z/VSE Java-capable platforms.

container. Is part of the JVM of application servers such as the IBM WebSphere Application Server, and facilitates the implementation of servlets, EJBs, and JSPs, by providing resource and transaction management resources. For example, an EJB developer must not code against the JVM of the application server, but instead against the interface that is provided by the container. The main role of a container is to act as an intermediary between EJBs and clients, Is the host part of the VSE JavaBeans, and is started using the job

STARTVCS, which is placed in the reader queue during the installation of z/VSE. Runs by default in dynamic class R. and also to manage multiple EJB instances. After EJBs have been written, they must be stored in a container residing on an application server. The container then manages all threading and client-interactions with the EJBs, and co-ordinate connection- and instance pooling.

control interval (CI). A fixed-length area of disk storage where VSE/VSAM stores records and distributes free space. It is the unit of information that VSE/VSAM transfers to or from disk storage. For FBA it must be an integral multiple to be defined at cluster definition, of the block size.

control program. A program to schedule and supervise the running of programs in a system.

conversational monitor system (CMS). A virtual machine operating system that provides general interactive time sharing, problem solving, and program development capabilities and operates under the control of z/VM.

count-key-data (CKD) device. A disk device that store data in the record format: count field, key field, data field. The count field contains, among others, the address of the record in the format: cylinder, head (track), record number, and the length of the data field. The key field, if present, contains the record's key or search argument. CKD disk space is allocated by tracks and cylinders. Contrast with *FBA disk device. See also extended count-key-data device.*

cross-partition communication control. A facility that enables VSE subsystems and user programs to communicate with each other; for example, with VSE/POWER.

cryptographic token. Usually referred to simply as a *token*, this is a device, which provides an interface for performing cryptographic functions like generating digital signatures or encrypting data.

cryptography.
1. The transformation of data to conceal its meaning.
2. In computer security, the principles, means, and methods for encrypting 'plaintext' and Is the host part of the VSE JavaBeans, and is started using the job STARTVCS, which is placed in the reader queue during installation of z/VSE. Runs by default in dynamic class R.decrypting 'ciphertext'.

D

data block group. The smallest unit of space that can be allocated to a VSE/POWER job on the data file. This allocation is independent of any device characteristics.

data conversion descriptor file (DCDF). With a DCDF, you can convert individual fields within a record during data transfer between a PC and its host. The DCDF defines the record fields of a particular file for both, the PC and the host environment.

data import. The process of reformatting data that was used under one operating system such that it can subsequently be used under a different operating system.

Data Interfile Transfer, Testing, and Operations (DITTO) utility. An IBM program that provides file-to-file services for card I/O, tape, and disk devices. The latest version is called DITTO/ESA for VSE.

Data Language/I (DL/I). A database access language that is used with CICS.

data link. In SNA, the combination of the link connection and the link stations joining network noes, for example, a z/Architecture channel and its associated protocols. A link is both logical and physical.

data security. Is the host part of the VSE JavaBeans, and is started using the job STARTVCS, which is placed in the reader queue during installation of z/VSE. Runs by default in dynamic class R. See *access control*.

data set header record. In VSE/POWER abbreviated as DSHR, alias NDH or DSH. An NJE control record either preceding output data or, in the middle of input data, indicating a change in the data format.

data space. A range of up to 2 gigabytes of contiguous virtual storage addresses that a program can directly manipulate through ESA/370 instructions. Unlike an address space, a data space can hold only user data; it does not contain shared areas, system data, or programs. Instructions do not execute in a data space, although in a program can reside in a data space as nonexecutable code. Contrast with address space.

data terminal equipment (DTE). In SNA, the part of a data station that serves a data source, data sink, or both.

database connector. Is a function introduced with z/VSE 5.1.1, which consists of a client and server part. The client provides an API (CBCLI) to be used by applications on z/VSE, the server on any Java capable platform connects a JDBC driver that is provided by the database. Both client and server communicate via TCP/IP.

Database 2 (DB2). An IBM rational database management system.

DB2-based connector. Is a feature introduced with VSE/ESA 2.5, which includes a customized DB2 version, together with VSAM and DL/I functionality, to provide access to DB2, VSAM, and DL/I data, using DB2 Stored Procedures.

DB2 Runtime only Client edition. The Client Edition for z/VSE comes with some enhanced features and improved performance to integrate z/VSE and Linux on System z.

DB2 Stored Procedure. In the context of z/VSE, a DB2 Stored Procedure is a Language Environment (LE) program that accesses DB2 data. However, from VSE/ESA 2.5 onwards you can also access VSAM and DL/I data using a DB2 Stored Procedure. In this way, it is possible to exchange data between VSAM and DB2.

DBLK. Data block.

DCDF. Data conversion descriptor file.

deblocking. The process of making each record of a block available for processing.

dedicated (disk) device. A device that cannot be shared among users.

device address.
1. The identification of an input/output device by its device number.
2. In data communication, the identification of any device to which data can be sent or from which data can be received.

device driving system (DDS). A software system external to VSE/POWER, such as a CICS spooler or PSF, that writes spooled output to a destination device.

Device Support Facilities (DSF). An IBM supplied system control program for performing operations on disk volumes so that they can be accessed by IBM and user programs. Examples of these operations are initializing a disk volume and assigning an alternative track.

device type code. The four- or five-digit code that is used for defining an I/O device to a computer system.

dialog. In an interactive system, a series of related inquiries and responses similar to a conversation between two people. For z/VSE, a set of panels that can be used to complete a specific task; for example, defining a file.

dialog manager. The program component of z/VSE that provides for ease of communication between user and system.

digital signature. In computer security, encrypted data, which is appended to or part of a message, that enables a recipient to prove the identity of the sender.

Digital Signature Algorithm (DSA). The Digital Signature Algorithm is the US government-defined standard for digital signatures. The DSA digital signature is a pair of large numbers, computed using a set of rules (that is, the DSA) and a set of parameters such that the identity of the signatory and integrity of the data can be verified. The DSA provides the capability to generate and verify signatures.

directory. In z/VSE the index for the program libraries.

direct access. Accessing data on a storage device using their address and not their sequence. This is the typical access on disk devices as opposed to magnetic tapes. Contrast with *sequential access*.

disk operating system residence volume (DORSES). The disk volume on which the system sublibrary IJSYSRS.SYSLIB is located including the programs and procedures that are required for system startup.

disk sharing. An option that lets independent computer systems uses common data on shared disk devices.

disposition. A means of indicating to VSE/POWER how a job input or output entry is to be handled: according to its local disposition in the RDR/LST/PUN queue or its transmission disposition when residing in the XMT queue. A job might, for example, be deleted or kept after processing.

distribution tape. A magnetic tape that contains, for example, a preconfigured operating system like z/VSE. This tape is shipped to the customer for program installation.

DITTO/ESA for VSE. Data Interfile Transfer, Testing, and Operations utility. An IBM program that provides file-to-file services for disk, tape, and card devices.

DSF. Device Support Facilities.

DSH (R). Data set header record.

dummy device. A device address with no real I/O device behind it. Input and output for that device address are spooled on disk.

duplex. Pertaining to communication in which data can be sent and received at the same time.

DU-AL (dispatchable unit - access list). The access list that is associated with a z/VSE main task or subtask. A program uses the DU-AL associated with its task and the PASN-AL associated with its partition. See also *PASN-AL*.

dynamic class table. Defines the characteristics of dynamic partitions.

dynamic partition. A partition that is created and activated on an 'as needed' basis that does not use fixed static allocations. After processing, the occupied space is released. Dynamic partitions are grouped by class, and jobs are scheduled by class. Contrast with *static partition*.

dynamic partition balancing. A z/VSE facility that allows the user to specify that two or more or all partitions of the system should receive about the same amount of time on the processor.

dynamic space reclamation. A librarian function that provides for space that is freed by the deletion of a library member to become reusable automatically.

E

ECI. See *CICS ECI.*

emulation. The use of programming techniques and special machine features that permit a computer system to execute programs that are written for another system or for the use of I/O devices different from those that are available.

emulation program (EP). An IBM control program that allows a channel-attached 3705 or 3725 communication controller to emulate the functions of an IBM 2701 Data Adapter Unit, or an IBM 2703 Transmission Control.

end user.
1. A person who makes use of an application program.
2. In SNA, the ultimate source or destination of user data flowing through an SNA network. Might be an application program or a terminal operator.

Enterprise Java Bean. An EJB is a distributed bean. "Distributed" means, that one part of an EJB runs inside the JVM of a web application server, while the other part runs inside the JVM of a web browser. An EJB either represents one data row in a database (entity bean), or a connection to a remote database (session bean). Normally, both types of an EJB work together. This allows to represent and access data in a standardized way in heterogeneous environments with relational and non-relational data. See also *JavaBean*.

entry-sequenced file. A VSE/VSAM file whose records are loaded without respect to their contents and whose relative byte addresses cannot change. Records are retrieved and stored by addressed access, and new records are added to the end of the file.

Environmental Record Editing and Printing (EREP) program. A z/VSE base program that makes the data that is contained in the system record file available for further analysis.

EPI. See *CICS EPI.*

ESCON Channel (Enterprise Systems Connection Channel). A serial channel, using fiber optic cabling, that provides a high-speed connection between host and control units for I/O devices. It complies with the ESA/390 and System z I/O Interface until z114. The zEC12 processors do not support ESCON channels.

exit routine.
1. Either of two types of routines: installation exit routines or user exit routines. Synonymous with exit program.
2. See *user exit routine*.

extended addressability. See *31 bit addressing*. The ability of a program to use virtual storage that is outside the address space in which the program is running. Generally, instructions and data reside in a single address space - the primary address space. However, a program can have data in address spaces other than the primary or in data spaces. (The instructions remain in the primary address space, while the data can reside in another address space, or in a data space.) To access data in other address spaces, a program must use access registers (ARs) and execute in access register mode (AR mode).

extended recovery facility (XRF). In z/VSE, a feature of CICS that provides for enhanced availability of CICS by offering one CICS system as a backup of another.

External Security Manager (ESM). A priced vendor product that can provide extended functionality and flexibility that is compared to that of the Basic Security Manager (BSM), which is part of z/VSE.

F

FASTCOPY. See *VSE/Fast Copy.*

fast copy data set program (VSE/Fast Copy). See *VSE/Fast Copy*.

fast service upgrade (FSU). A service function of z/VSE for the installation of a refresh release without regenerating control information such as library control tables.

FAT-DASD. A subtype of Large DASD, it supports a device with more than 4369 cylinders (64 K tracks) up to 64 K cylinders.

FCOPY. See *VSE/Fast Copy.*

fence. A separation of one or more components or elements from the remainder of a processor complex. The separation is by logical boundaries. It allows simultaneous user operations and maintenance procedures.

fetch.
1. To locate and load a quantity of data from storage.
2. To bring a program phase into virtual storage from a sublibrary and pass control to this phase.
3. The name of the macro instruction (FETCH) used to accomplish 2. See also *loader.*

Fibre Channel Protocol (FCP). A combination of hardware and software conforming to the Fibre Channel standards and allowing system and peripheral connections via FICON and FICON Express feature cards on IBM zSeries processors. In z/VSE, zSeries FCP is employed to access industry-standard SCSI disk devices.

fragmentation (of storage). Inability to allocate unused sections (fragments) of storage in the real or virtual address range of virtual storage.

FSU. Fast service upgrade.

FULIST (FUnction LIST). A type of selection panel that displays a set of files and/or functions for the choice of the user.

G

generation. See *macro generation.*

generation feature. An IBM licensed program order option that is used to tailer the object code of a program to user requirements.

GETVIS space. Storage space within partition or the shared virtual area, available for dynamic allocation to programs.

guest system. A data processing system that runs under control of another (host) system. On the mainframe z/VSE can run as a guest of z/VM.

H

hard wait. The condition of a processor when all operations are suspended. System recovery from a hard wait is impossible without performing a new system startup.

hash function. A hash function is a transformation that takes a variable-size input and returns a fixed-size string, which is called the hash value. In cryptography, the hash functions should have some additional properties:
- The hash function should be easy to compute.
- The hash function is one way; that is, it is impossible to calculate the 'inverse' function.
- The hash function is collision-free; that is, it is impossible that different input leads to the same hash value.

hash value. The fixed-sized string resulting after applying a *hash function* to a text.

High-Level Assembler for VSE. A programming language providing enhanced assembler programming support. It is a base program of z/VSE.

home interface. Provides the methods to instantiate a new EJB object, introspect an EJB, and remove an EJB instantiation., as for the remote interface is needed because the deployment tool generates the implementation class. Every Session bean's home interface must supply at least one *create()* method.

host mode. In this operating mode, a PC can access a VSE host. For programmable workstation (PWS) functions, the Move Utilities of VSE can be used.

host system. The controlling or highest level system in a data communication configuration.

host transfer file (HTF). Used by the Workstation File Transfer Support of z/VSE as an intermediate storage area for files that are sent to and from IBM personal computers.

HTTP Session. In the context of z/VSE, identifies the web-browser client that calls a servlet (in other words, identifies the connection between the client and the middle-tier platform).

I

ICCF. See *VSE/ICCF.*

ICKDSF (Device Support Facilities). A z/VSE base program that supports the installation, use, and maintenance of IBM disk devices.

include function. Retrieves a library member for inclusion in program input.

index.
1. A table that is used to locate records in an indexed sequential data set or on indexed file.
2. In, an ordered collection of pairs, each consisting of a key and a pointer, used by to sequence and locate the records of a key-sequenced data set or file; it is organized in levels of index records. See also *alternate index.*

input/output control system (IOCS). A group of IBM supplied routines that handle the transfer of data between main storage and auxiliary storage devices.

integrated communication adapter (ICA). The part of a processor where multiple lines can be connected.

integrated console. In z/VSE, the service processor console available on IBM System z server that operates as the z/VSE system console. The integrated console is typically used during IPL and for recovery purposes when no other console is available.

Interactive Computing and Control Facility (ICCF). An IBM licensed program that serves as interface, on a time-slice basis, to authorized users of terminals that are linked to the system's processor.

interactive partition. An area of virtual storage for the purpose of processing a job that was submitted interactively via VSE/ICCF.

Interactive User Communication Vehicle (IUCV). Programming support available in a VSE supervisor for operation under z/VM. The support allows users to communicate with other users or with CP in the same way they would with a non-preferred guest.

intermediate storage. Any storage device that is used to hold data temporarily before it is processed.

IOCS. Input/output control system.

IPL. Initial program load.

irrecoverable error. An error for which recovery is impossible without the use of recovery techniques external to the computer program or run.

IUCV. Interactive User Communication Vehicle.

J

JAR. Is a platform-independent file format that aggregates many files into one. Multiple applets and their requisite components (.class files, images, and sounds) can be bundled in a JAR file, and then downloaded to a web browser using a single HTTP transaction (much improving the download speed). The JAR format also supports compression, which reduces the files size (and further improves the download speed). The compression algorithm that is used is fully compatible with the ZIP algorithm. The owner of an applet can also digitally sign individual entries in a JAR file to authenticate their origin.

Java application. A Java program that runs inside the JVM of your web browser. The program's code resides on a local hard disk or on the LAN. Java applications might be large programs using graphical interfaces. Java applications have unlimited access to all your local resources.

Java bytecode. Bytecode is created when a file containing Java source language statements is compiled. The compiled Java code or "bytecode" is similar to any program module or file that is ready to be executed (run on a computer so that instructions are performed one at a time). However, the instructions in the bytecode are really instructions to the *Java Virtual Machine*. Instead of being interpreted one instruction at a time, bytecode is instead recompiled for each operating-system platform using a just-in-time (JIT) compiler. Usually, this enables the Java program to run faster. Bytecode is contained in binary files that have the suffix**.CLASS**

Java servlet. See *servlet.*

JHR. Job header record.

job accounting interface. A function that accumulates accounting information for each job step, to be used for charging the users of the system, for planning new applications, and for supervising system operation more efficiently.

job accounting table. An area in the supervisor where accounting information is accumulated for the user.

job catalog. A catalog made available for a job by means of the file name IJSYSUC in the respective DLBL statement.

job entry control language (JECL). A control language that allows the programmer to specify how VSE/POWER should handle a job.

job step. In 1 of a group of related programs complete with the JCL statements necessary for a particular run. Every job step is identified in the job stream by an EXEC statement under one JOB statement for the whole job.

job trailer record (JTR). As VSE/POWER parameter JTR, alias NJT. An NJE control record terminating a job entry in the input or output queue and providing accounting information.

K

key. In VSE/VSAM, one or several characters that are taken from a certain field (key field) in data records for identification and sequence of index entries or of the records themselves.

key sequence. The collating sequence either of records themselves or of their keys in the index or both. The key sequence is alphanumeric.

key-sequenced file. A VSE/VSAM file whose records are loaded in key sequence and controlled by an index. Records are retrieved and stored by keyed access or by addressed access, and new records are inserted in the file in key sequence.

KSDS. Key-sequenced data sets. See *key-sequenced file.*

L

label.
1. An identification record for a tape, disk, or diskette volume or for a file on such a volume.
2. In assembly language programming, a named instruction that is generally used for branching.

label information area. An area on a disk to store label information that is read from job control statements or commands. Synonymous with *label area*.

Language Environment for z/VSE. An IBM software product that is the implementation of Language Environment on the VSE platform.

language translator. A general term for any assembler, compiler, or other routine that accepts statements in one language and produces equivalent statements in another language.

Large DASD. A DASD device that
1. Has a capacity exceeding 64 K tracks and
2. Does not have VSAM space created prior to VSE/ESA 2.6 that is owned by a catalog.

LE/VSE. Short form of Language Environment for z/VSE.

librarian. The set of programs that maintains, services, and organizes the system and private libraries.

library block. A block of data that is stored in a sublibrary.

library directory. The index that enables the system to locate a certain sublibrary of the accessed library.

library member. The smallest unit of a data that can be stored in and retrieved from a sublibrary.

line commands. In VSE/ICCF, special commands to change the declaration of individual lines on your screen. You can copy, move, or delete a line declaration, for example.

linkage editor. A program that is used to create a phase (executable code) from one or more independently translated object modules, from one or more existing phases, or from both. In creating the phase, the linkage editor resolves cross-references among the modules and phases available as input. The program can catalog the newly built phases.

linkage stack. An area of protected storage that the system gives to a program to save status information in a branch or a program call.

link station. In SNA, the combination of hardware and software that allows a node to attach to and provide control for a link.

loader. A routine, commonly a computer program, that reads data or a program into processor storage. See also *relocating loader*.

local shared resources (LSR). A VSE/VSAM option that is activated by three extra macros to share control blocks among files.

lock file. In a shared disk environment under VSE, a system file on disk that is used by the sharing systems to control their access to shared data.

logical partition. In LPAR mode, a subset of the server unit hardware that is defined to support the operation of a system control program.

logical record. A user record, normally pertaining to a single subject and processed by data management as a unit. Contrast with *physical* record, which may be larger or smaller.

logical unit (LU).
1. A name that is used in programming to represent an I/O device address. *physical unit (PU), system services control point (SSCP), primary logical unit (PLU), and secondary logical unit (SLU).*
2. In SNA, a port through which a user accesses the SNA network,
   a. To communicate with another user and
   b. To access the functions of the SSCP. An LU can support at least two sessions. One with an SSCP and one with another LU and might be capable of supporting many sessions with other LUs.

logical unit name. In programming, a name that is used to represent the address of an input/output unit.

logical unit 6.2. A SNA/SDLC protocol for communication between programs in a distributed processing environment. LU 6.2 is characterized by
1. A peer relationship between session partners,
2. Efficient utilization of a session for multiple transactions,
3. Comprehensive end-to-end error processing, and
4. A generic Application Programming Interface (API) consisting of structured verbs that are mapped into a product implementation.

logons interpret interpret routine. In VTAM, an installation exit routine, which is associated with an interpret table entry, that translates logon information. It also verifies the logon.

LPAR mode. Logically partitioned mode. The CP mode that is available on the Configuration (CONFIG) frame when the PR/SM feature is installed. LPAR mode allows the operator to allocate the hardware resources of the processor unit among several logical partitions.

M

macro definition. A set of statements and instructions that defines the name of, format of, and conditions for generating a sequence of assembler statements and machine instructions from a single source statement.

macro expansion. See *macro generation*

macro generation. An assembler operation by which a macro instruction gets replaced in the program by the statements of its definition. It takes place before assembly. Synonymous with *macro expansion*.

macro (instruction).
1. In assembler programming, a user-invented assembler statement that causes the assembler to

process a set of statements that are defined previously in the macro definition.
2. A sequence of VSE/ICCF commands that are defined to cause a sequence of certain actions to be performed in response to one request.

maintain system history program (MSHP). A program that is used for automating and controlling various installation, tailoring, and service activities for a VSE system.

main task. The main program within a partition in a multiprogramming environment.

master console. In z/VSE, one or more consoles that receive all system messages, except for those that are directed to one particular console. Contrast this with the *user* console, which receives only those messages that are specifically directed to it, for example messages that are issued from a job that was submitted with the request to echo its messages to that console. The operator of a master console can reply to all outstanding messages and enter all system commands.

maximum (max) CA. A unit of allocation equivalent to the maximum control area size on a count-key-data or fixed-block device. On a CKD device, the max CA is equal to one cylinder.

memory object. Chunk of virtual storage that is allocated above the bar (2 GB) to be created with the IARV64 macro.

message. In VSE, a communication that is sent from a program to the operator or user. It can appear on a console, a display terminal or on a printout.

MSHP. See maintain system history program.

multitasking. Concurrent running of one main task and one or several subtasks in the same partition.

MVS. Multiple Virtual Storage. Implies MVS/390, MVS/XA, MVS/ESA, and the MVS element of the z/OS (OS/390) operating system.

N

NetView. A z/VSE optional program that is used to monitor a network, manage it, and diagnose its problems.

network address. In SNA, an address, consisting of subarea and element fields, that identifies a link, link station, or NAU. Subarea nodes use network addresses; peripheral nodes use local addresses. The boundary function in the subarea node to which a peripheral node is attached transforms local addresses to network addresses and vice versa. See also *network name*.

network addressable unit (NAU). In SNA, a logical unit, a physical unit, or a system services control point. It is the origin or the destination of information that is transmitted by the path control network. Each NAU has a network address that represents it to the path control network. See also *network name, network address*.

Network Control Program (NCP). An IBM licensed program that provides communication controller support for single-domain, multiple-domain, and interconnected network capability. Its full name is ACF/NCP.

network definition table (NDT). In VSE/POWER networking, the table where every node in the network is listed.

network name.
1. In SNA, the symbolic identifier by which users refer to a NAU, link, or link station. See also *network address*.
2. In a multiple-domain network, the name of the APPL statement defining a VTAM application program. This is its network name, which must be unique across domains.

node.
1. In SNA, an end point of a link or junction common to several links in a network. Nodes can be distributed to host processors, communication controllers, cluster controllers, or terminals. Nodes can vary in routing and other functional capabilities.
2. In VTAM, a point in a network that is defined by a symbolic name. Synonymous with *network node*. See *major node and minor node*.

node type. In SNA, a designation of a node according to the protocols it supports and the network addressable units (NAUs) it can contain.

O

object module (program). A program unit that is the output of an assembler or compiler and is input to a linkage editor.

online application program. An interactive program that is used at display stations. When active, it waits for data. Once input arrives, it processes it and send a response to the display station or to another device.

operator command. A statement to a control program, issued via a console or terminal. It causes the control program to provide requested information, alter normal operations, initiate new operations, or end existing operations.

optional licensed program. An IBM licensed program that a user can install on VSE by way of available installation-assist support.

output parameter text block (OPTB). in VSE/POWER's spool-access support, information that is contained in an output queue record if a * $$ LST or * $$ PUN statement includes any user-defined keywords that have been defined for autostart.

P

page data set (PDS). One or more extents of disk storage in which pages are stored when they are not needed in processor storage.

page fixing. Marking a page so that it is held in processor storage until explicitly released. Until then, it cannot be paged out.

page I/O. Page-in and page-out operations.

page pool. The set of page frames available for paging virtual-mode programs.

panel. The complete set of information that is shown in a single display on terminal screen. Scrolling back and forth through panels like turning manual pages. See also *selection panel*.

partition balancing, dynamic. A z/VSE facility that allows the user to specify that two or more or all partitions of the system should receive about the same amount of time on the processor.

PASN-AL (primary address space number - access list). The access list that is associated with a partition. A program uses the PASN-AL associated with its partition and the DU-AL associated with its task (work unit). See also *DU-AL*.

Each partition has its own unique PASN-AL. All programs running in this partition can access data spaces through the PASN-AL. Thus a program can create a data space, add an entry for it in the PASN-AL, and obtain the ALET that indexes the entry. By passing the ALET to other programs in the partition, the program can share the data space with other programs running in the same partition.

PDS. Page data sets.

phase. The smallest complete unit of executable code that can be loaded into virtual storage.

physical record. The amount of data that is transferred to or from auxiliary storage. Synonymous with *block*.

PNET. Programming support available with VSE/POWER; it provides for the transmission of selected jobs, operator commands, messages, and program output between the nodes of a network.

POWER. See *VSE/POWER*.

pregenerated operating system. An operating system such as z/VSE that is shipped by IBM mainly in object code. IBM defines such key characteristics as the size of the main control program, the organization, and size of libraries, and required system areas on disk. The customer does not have to generate an operating system.

preventive service. The installation of one or more PTFs on a VSE system to avoid the occurrence of anticipated problems.

primary address space. In z/VSE, the address space where a partition is executed. A program in primary mode fetches data from the primary address space.

primary library. A VSE library owned and directly accessible by a certain terminal user.

printer/keyboard mode. Refers to 1050 or 3215 console mode (device dependent).

Print Services Facility (PSF)/VSE. An access method that provides support for the advanced function printers.

private area. The virtual space between the shared area (24 bit) and shared area (31 bit), where (private) partitions are allocated. Its maximum size can be defined during IPL. See also *shared area*.

private memory object. Memory object (chunk of virtual storage) that is allocated above the 2 GB line (bar) only accessible by the partition that created it.

private partition. Any of the system's partitions that are not defined as shared. See also *shared partition*.

production library.
1. In a pre-generated operating system (or product), the program library that contains the object code for this system (or product).
2. A library that contains data that is needed for normal processing. Contrast with *test library*.

programmer logical unit. A logical unit available primarily for user-written programs. See also *logical unit name*.

program temporary fix (PTF). A solution or by-pass of one or more problems that are documented in APARs. PTFs are distributed to IBM customers for preventive service to a current release of a program.

PSF/VSE. Print Services Facility/VSE.

PTF. See *Program temporary fix*.

Q

Queue Control Area (QCA). In VSE/POWER, an area of the data file, which might contain:
- Extended checkpoint information
- Control information for a shared environment.

queue file. A direct-access file that is maintained by VSE/POWER that holds control information for the spooling of job input and job output.

R

random processing. The treatment of data without respect to its location on disk storage, and in an arbitrary sequence that is governed by the input against which it is to be processed.

real address area. In z/VSE, processor storage to be accessed with dynamic address translation (DAT) off

real address space. The address space whose addresses map one-to-one to the addresses in processor storage.

real mode. In VSE, a processing mode in which a program might not be paged. Contrast with *virtual mode*.

recovery management support (RMS). System routines that gather information about hardware failures and that initiate a retry of an operation that failed because of processor, I/O device, or channel errors.

refresh release. An upgraded VSE system with the latest level of maintenance for a release.

relative-record file. A VSE/VSAM file whose records are loaded into fixed-length slots and accessed by the relative-record numbers of these slots.

release upgrade. Use of the FSU functions to install a new release of z/VSE.

relocatable module. A library member of the type object. It consists of one or more control sections cataloged as one member.

relocating loader. A function that modifies addresses of a phase, if necessary, and loads the phase for running into the partition that is selected by the user.

remote interface. In the context of z/VSE, the remote interface allows a client to make method calls to an EJB although the EJB is on a remote z/VSE host. The container uses the remote interface to create client-side stubs and server-side proxy objects to handle incoming method calls from a client to an EJB.

remote procedure call (RPC).

1. A facility that a client uses to request the execution of a procedure call from a server. This facility includes a library of procedures and an external data representation.
2. A client request to service provider in another node.

residency mode (RMODE). A program attribute that refers to the location where a program is expected to reside in virtual storage. RMODE 24 indicates that the program must reside in the 24-bit addressable area (below 16 megabytes), RMODE ANY indicates that the program can reside anywhere in 31-bit addressable storage (above or below 16 megabytes).

REXX/VSE. A general-purpose programming language, which is particularly suitable for command procedures, rapid batch program development, prototyping, and personal utilities.

RMS. Recovery management support.

RPG II. A commercially oriented programming language that is specifically designed for writing application programs that are intended for business data processing.

S

SAM ESDS file. A SAM file that is managed in VSE/VSAM space, so it can be accessed by both SAM and VSE/VSAM macros.

SCP. System control programming.

SDL. System directory list.

search chain. The order in which chained sublibraries are searched for the retrieval of a certain library member of a specified type.

second-level directory. A table in the SVA containing the highest phase names that are found on the directory tracks of the system sublibrary.

Secure Sockets Layer (SSL). A security protocol that allows the client to authenticate the server and all data and requests to be encrypted. SSL was developed by Netscape Communications Corp. and RSA Data Security, Inc..

segmentation. In VSE/POWER, a facility that breaks list or punch output of a program into segments so that printing or punching can start before this program has finished generating such output.

selection panel. A displayed list of items from which a user can make a selection. Synonymous with *menu*.

sense. Determine, on request or automatically, the status or the characteristics of a certain I/O or communication device.

sequential access method (SAM). A data access method that writes to and reads from an I/O device record after record (or block after block). On request, the support performs device control operations such as line spacing or page ejects on a printer or skip some tape marks on a tape drive.

service node. Within the VSE unattended node support, a processor that is used to install and test a master VSE system, which is copied for distribution to the unattended nodes. Also, program fixes are first applied at the service node and then sent to the unattended nodes.

service program. A computer program that performs function in support of the system. See with *utility program*.

service refresh. A form of service containing the current version of all software. Also referred to as a *system refresh*.

service unit. One or more PTFs on disk or tape (cartridge).

shared area. In z/VSE, shared areas (24 bit) contain the Supervisor areas and SVA (24 bit) and shared areas (31 bit) the SVA (31 bit). Shared areas (24 bit) are at the beginning of the address space (below 16 MB), shared area (31 bit) at the end (below 2 GB).

shared disk option. An option that lets independent computer systems use common data on shared disk devices.

shared memory objects. An option that lets independent computer systems uses common data on shared disk devices.

shared partition. In z/VSE, a partition that is allocated for a program (VSE/POWER, for example) that provides services and communicates with programs in other partitions of the system's virtual address spaces.

shared spooling. A function that permits the VSE/POWER account file, data file, and queue file to be shared among several computer systems with VSE/POWER.

shared virtual area (SVA). In z/VSE, a high address area that contains a list system directory list (SDL) of frequently used phases, resident programs that are shared between partitions, and an area for system support.

SIT (System Initialization Table). A table in CICS that contains data used the system initialization process. In particular, the SIT can identify (by suffix characters) the version of CICS system control programs and CICS tables that you have specified and that are to be loaded.

skeleton. A set of control statements, instructions, or both, that requires user-specific information to be inserted before it can be submitted for processing.

socksified. See *socks-enabled*.

Socks-enabled. Pertaining to TCP/IP software, or to a specific TCP/IP application, that understands the *socks protocol*. "Socksified" is a slang term for socks-enabled.

socks protocol. A protocol that enables an application in a secure network to communicate through a firewall via a *socks server*.

socks server. A circuit-level gateway that provides a secure one-way connection through a firewall to server applications in a nonsecure network.

source member. A library member containing source statements in any of the programming languages that are supported by VSE.

split. To double a specific unit of storage space (CI or CA) dynamically when the specified minimum of free space gets used up by new records.

spooling. The use of disk storage as buffer storage to reduce processing delays when transferring data between peripheral equipment and the processor of a computer. In z/VSE, this is done under the control of VSE/POWER.

Spool Access Protection. An optional feature of VSE/POWER that restricts individual spool file entry access to user IDs that have been authenticated by having performed a security logon.

spool file.
1. A file that contains output data that is saved for later processing.
2. One of three VSE/POWER files on disk: queue file, data file, and account file.

stacked tape. An IBM supplied product-shipment tape containing the code of several licensed programs.

standard label. A fixed-format record that identifies a volume of data such as a tape reel or a file that is part of a volume of data.

stand-alone program. A program that runs independently of (not controlled by) the VSE system.

startup. The process of performing IPL of the operating system and of getting all subsystems and applications programs ready for operation.

start option. In VTAM, a user-specified or IBM specified option that determines conditions for the time a VTAM system is operating. Start options can be predefined or specified when VTAM is started.

static partition. A partition, which is defined at IPL time and occupying a defined amount of virtual storage that remains constant. See also *dynamic partition*.

storage director. An independent component of a storage control unit; it performs all of the functions of a storage control unit and thus provides one access path to the disk devices that are attached to it. A storage control unit has two storage directors.

storage fragmentation. Inability to allocate unused sections (fragments) of storage in the real or virtual address range of virtual storage.

suballocated file. A VSE/VSAM file that occupies a portion of an already defined data space. The data space might contain other files. See also *unique file*.

sublibrary. In VSE, a subdivision of a library. Members can only be accessed in a sublibrary.

sublibrary directory. An index for the system to locate a member in the accessed sublibrary.

submit. A VSE/POWER function that passes a job to the system for processing.

SVA. See shared virtual area.

Synchronous DataLink Control (SDLC). A discipline for managing synchronous, code-transparent, serial-by-bit information transfer over a link connection. Transmission exchanges might be duplex or half-duplex over switched or non-switched links. The configuration of the link connection might be point-to-point, multipoint, or loop.

SYSRES. See system residence volume.

system control programming (SCP). IBM supplied, non-licensed program fundamental to the operation of a system or to its service or both.

system directory list (SDL). A list containing directory entries of frequently used phases and of all phases resident in the SVA. The list resides in the SVA.

system file. In z/VSE, a file that is used by the operating system, for example, the hardcopy file, the recorder file, the page data set.

System Initialization Table (SIT). A table in CICS that contains data that is used by the system initialization process. In particular, the SIT can identify (by suffix characters) the version of CICS system control programs and CICS tables that you have specified and that are to be loaded.

system recorder file. The file that is used to record hardware reliability data. Synonymous with *recorder file*.

system refresh. See *service refresh*.

system refresh release. See *refresh release*.

system residence file (SYSRES). The z/VSE system sublibrary IJSYSRS.SYSLIB that contains the operating system. It is stored on the system residence volume DORSES.

system residence volume (SYSRES). The disk volume on which the system sublibrary is stored and from which the hardware retrieves the initial program load routine for system startup.

system sublibrary. The sublibrary that contains the operating system. It is stored on the system residence volume (SYSRES).

T

task management. The functions of a control program that control the use, by tasks, of the processor and other resources (except for input/output devices).

time event scheduling support. In VSE/POWER, the time event scheduling support offers the possibility to schedule jobs for processing in a partition at a predefined time once repetitively. The time event scheduling operands of the * $$ JOB statement are used to specify the wanted scheduling time.

track group. In VSE/POWER, the basic organizational unit of a file for CKD devices.

track hold. A function that protects a track that is being updated by one program from being accessed by another program.

transaction.

1. In a batch or remote batch entry, a job or job step. 2. In CICS TS, one or more application programs that can be used by a display station operator. A given transaction can be used concurrently from one or more display stations. The execution of a transaction for a certain operator is also referred to as a task.
2. A given task can relate only to one operator.

transient area. An area within the control program that is used to provide high-priority system services on demand.

Turbo Dispatcher. A facility of z/VSE that allows to use multiprocessor systems (also called CEC: Central Electronic Complexes). Each CPU within such a CEC has accesses to be shared virtual areas of z/VSE: supervisor, shared areas (24 bit), and shared areas (31 bit). The CPUs have equal rights, which means that any CPU might receive interrupts and work units are not dedicated to any specific CPU.

U

UCB. Universal character set buffer.

universal character set buffer (UCB). A buffer to hold UCS information.

user console. In z/VSE, a console that receives only those system messages that are specifically directed to it. These are, for example, messages that are issued from a job that was submitted with the request to echo its messages to that console. Contrast with *master console*.

user exit. A programming service that is provided by an IBM software product that can be requested during the execution of an application program for the service of transferring control back to the application program upon the later occurrence of a user-specified event.

V

variable-length relative-record data set (VRDS). A relative-record data set with variable-length records. See also *relative-record data set*.

variable-length relative-record file. A VSE/VSAM relative-record file with variable-length records. See also *relative-record file*.

VIO. See virtual I/O area.

virtual address. An address that refers to a location in virtual storage. It is translated by the system to a processor storage address when the information stored at the virtual address is to be used.

virtual addressability extension (VAE). A storage management support that fives the user of VSE multiple address spaces of virtual storage.

virtual address space. A subdivision of the virtual address area available to the user for the allocation of private, nonshared partitions.

virtual disk. A range of up to 2 gigabytes of contiguous virtual storage addresses that a program can use as workspace. Although the virtual disk exists in storage, it appears as a real FBA disk device to the user program. All I/O operations that are directed to a virtual disk are intercepted and the data to be written to, or read from, the disk is moved to or from a data space.

Like a data space, a virtual disk can hold only user data; it does not contain shared areas, system data, or programs. Unlike an address space or a data space, data is not directly addressable on a virtual disk. To manipulate data on a virtual disk, the program must perform I/O operations.

virtual I/O area (VIO). An extension of the page data set; used by the system as intermediate storage, primarily for control data.

virtual mode. The operating mode of a program can be paged.

virtual partition. In VSE, a division of the dynamic area of virtual storage.

virtual storage. Addressable space image for the user from which instructions and data are mapped into processor storage locations.

virtual tape. In z/VSE, a virtual tape is a file (or data set) containing a tape image. You can read from or write to a virtual tape in the same way as if it were a physical tape. A virtual tape can be:
- A VSE/VSAM ESDS file on the z/VSE host side.
- A remote file on the server side; for example, a Linux, UNIX, or Windows file. To access such a remote virtual tape, a TCP/IP connection is required between z/VSE and the remote system.

volume ID. The volume serial number, which is a number in a volume label that is assigned when a volume is prepared for use by the system.

VRDS. Variable-length relative-record data sets. See *variable-length relative record file*.

VSAM. See *VSE/VSAM*.

VSE (Virtual Storage Extended). A system that consists of a basic operating system and any IBM supplied and user-written programs that are required to meet the data processing needs of a user. VSE and hardware it controls form a complete computing system. Its current version is called z/VSE.

VSE/Advanced Functions. As part of VSE Central Functions, a base program of z/VSE. A program that provides basic system control and includes the supervisor and system programs such as the Librarian and the Linkage Editor.

VSE Connector Server. Is the host part of the VSE JavaBeans, and is started using the job STARTVCS, which is placed in the reader queue during installation of z/VSE. Runs by default in dynamic class R.

VSE/DITTO (VSE/Data Interfile Transfer, Testing, and Operations Utility). An IBM licensed program that provides file-to-file services for disk, tape, and card devices.

VSE/ESA (Virtual Storage Extended/Enterprise Systems Architecture). The predecessor system of z/VSE.

VSE/Fast Copy. A utility program for fast copy data operations from disk to disk and dump/restore operations via an intermediate dump file on magnetic tape or disk.

VSE/FCOPY (VSE/Fast Copy Data Set program). An IBM licensed program for fast copy data operations from disk to disk and dump/restore operations via an intermediate dump file on magnetic tape or disk. There is also a stand-alone version: the FASTCOPY utility.

VSE/ICCF (VSE/Interactive Computing and Control Facility). An IBM licensed program that serves as interface, on a time-slice basis, to authorized users of terminals that are linked to the system's processor.

VSE/ICCF library. A file that is composed of smaller files (libraries) including system and user data, which can be accessed under the control of VSE/ICCF.

VSE JavaBeans. Are JavaBeans that allow access to all VSE-based file systems (VSE/VSAM, Librarian, and VSE/ICCF), submit jobs, and access the z/VSE operator console. The class library is contained in the *VSEConnector.jar* archive. See also *JavaBeans*.

VSE library. A collection of programs in various forms and storage dumps stored on disk. The form of a program is indicated by its member type such as source code, object module, phase, or procedure. A VSE library consists of at least one sublibrary, which can contain any type of member.

VSE/POWER. An IBM licensed program that is primarily used to spool input and output. The program's networking functions enable a VSE system to exchange files with or run jobs on another remote processor.

VSE/VSAM (VSE/Virtual Storage Access Method). An IBM access method for direct or sequential processing of fixed and variable length records on disk devices.

VSE/VSAM catalog. A file containing extensive file and volume information that VSE/VSAM requires to locate files, to allocate and deallocate storage space, to verify the authorization of a program or an operator to gain access to a file, and to accumulate use statistics for files.

VSE/VSAM managed space. A user-defined space on disk that is placed under the control of VSE/VSAM.

W

wait for run subqueue. In VSE/POWER, a subqueue of the reader queue with dispatchable jobs ordered in execution start time sequence.

wait state. The condition of a processor when all operations are suspended. System recovery from a hard wait is impossible without performing a new system startup. See *hard wait*.

Workstation File Transfer Support. Enables the exchange of data between IBM Personal Computers (PCs) linked to a z/VSE host system where the data is kept in intermediate storage. PC users can retrieve that data and work with it independently of z/VSE.

work file. A file that is used for temporary storage of data being processed.

Numerics

24-bit addressing. Provides addressability for address spaces up to 16 megabytes.

31-bit addressing. Provides addressability for address spaces up to 2 gigabytes.

64-bit addressing. Provides addressability for address spaces up to 2 gigabytes and above. See also *24-bit addressing*.

Index

Special characters

Numerics

A

B

C

D

E

Q

R

S

T

U

V

X

Z

Readers' Comments — We'd Like to Hear from You

IBM z/VSE
VSE Central Functions
VSE/POWER Application Programming
Version 9 Release 2

Publication No. SC34-2642-01

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:
- Send your comments to the address on the reverse side of this form.
- Send a fax to the following number: +49-7031-163456
- Send your comments via email to: s390id@de.ibm.com
- Send a note from the web page: http://www.ibm.com/systems/z/os/zvse/

If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	Email address

Readers' Comments — We'd Like to Hear from You
SC34-2642-01

IBM®

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Deutschland Research & Development GmbH
Department 3282
Schoenaicher Strasse 220
71032 Boeblingen
Germany

Fold and Tape **Please do not staple** Fold and Tape

SC34-2642-01

Cut or Fold
Along Line

Product Number: 5686-CF9

Printed in USA

SC34-2642-01