z/OS



JES Application Programming

SA23-2240-01

z/OS



JES Application Programming

SA23-2240-01

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 45.

Second Edition, September 2009

This edition applies to Version 1 Release 11 of z/OS (5694-A01) until otherwise indicated in new editions.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this document, or you may address your comments to the following address:

International Business Machines Corporation
MHVRCFS, Mail Station P181
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+845+432-9405
FAX (Other Countries):
Your International Access Code +1+845+432-9405

IBMLink™ (United States customers only): IBMUSM10(MHVRCFS)
Internet e-mail: mhvrcfs@us.ibm.com
World Wide Web: http://www.ibm.com/systems/z/os/zos/webqs.html

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this document
- Page number or topic related to your comment

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

© Copyright International Business Machines Corporation 2008, 2009.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures v

Tables vii

About this document ix
Who should use this document ix
How to use this document ix
Where to Find More Information ix
The z/OS Basic Skills Information Center ix

Summary of changes xi

Chapter 1. Introduction 1

Chapter 2. JES Spool Data Set Browse 3
Allocation 3
Using the Compatibility Interface 7
Using the ACB/RPL Interface 8
End of File Processing 12
Performance 12
Secondary Subsystem Support 13

Chapter 3. JES Client/Server Print Interface 15
Creating a CTOKEN 15
Comparing CTOKENs 15
Obtaining Status for a Data Set 16
Identifying a Requestor on a Header Page 17
Listening for Events 18

Chapter 4. Internal Reader Facility 21
Defining the internal reader facility 21
Using the internal reader facility 21
JES control statements that affect the internal reader 24
Performance considerations for JES internal reader 24

Chapter 5. The External Writer 27
Overview of the IBM-Supplied External Writer 27
How to Set Up and Start the External Writer 28
How the IBM-Supplied Output Writer Routine Works 32
How to Add Your Own Output Writing Routine 34

Appendix. Accessibility 43
Using assistive technologies 43
Keyboard navigation of the user interface 43
z/OS information 43

Notices 45
Programming Interface Information 46
Policy for unsupported hardware 47
Trademarks 47

Index 49

© Copyright IBM Corp. 2008, 2009

Figures

1. Submitting a Job to the Internal Reader 22
2. The RDR Procedure 23
3. General Flow of IBM's Output Writing Routine 34

© Copyright IBM Corp. 2008, 2009

Tables

1. Additional RBA formats 10
2. External Writer Parameter List 32
3. Separator Routine Parameter List. 39
4. Block Character Routine Parameter List 41

© Copyright IBM Corp. 2008, 2009

About this document

This document supports z/OS (5694-A01).

This document describes the application programming of JES2 and JES3. It provides the information that you need to:

- Use the Spool Data Set Browse
- Use the Server/Client Print Interface
- Use the Internal Reader
- Use the IBM® supplied External Writer

Who should use this document

This document is intended for JES2 and JES3 application programmers who are using spool data set browse, server/client print interface, internal reader, and external writer.

How to use this document

Use this document in conjunction with the following documents:

- *z/OS JES2 Initialization and Tuning Guide*
- *z/OS JES3 Initialization and Tuning Guide*
- *z/OS MVS Using the Subsystem Interface*

Most referenced publications are abbreviated throughout the text; their full titles appear in “Where to Find More Information” on page ix, which follows.

Where to Find More Information

This document references the following publications for further details about specific topics. Abbreviated forms of these titles are used throughout this document. The following table lists all abbreviated titles, full titles, and their order numbers that are not listed in the z/OS® Information Roadmap. See that document for all z/OS publications.

Short Title Used in This document	Title	Order Number
JES2 Initialization and Tuning Guide	*z/OS JES2 Initialization and Tuning Guide*	SA22-7532
JES3 Initialization and Tuning Guide	*z/OS JES3 Initialization and Tuning Guide*	SA22-7549
MVS Using the Subsystem Interface	*z/OS MVS Using the Subsystem Interface*	SA22-7642

The z/OS Basic Skills Information Center

The z/OS Basic Skills Information Center is a Web-based information resource intended to help users learn the basic concepts of z/OS, the operating system that runs most of the IBM mainframe computers in use today. The Information Center is designed to introduce a new generation of Information Technology professionals to basic concepts and help them prepare for a career as a z/OS professional, such as a z/OS system programmer.

© Copyright IBM Corp. 2008, 2009

Specifically, the z/OS Basic Skills Information Center is intended to achieve the following objectives:

- Provide basic education and information about z/OS without charge
- Shorten the time it takes for people to become productive on the mainframe
- Make it easier for new people to learn z/OS.

To access the z/OS Basic Skills Information Center, open your Web browser to the following Web site, which is available to all users (no login required): http://publib.boulder.ibm.com/infocenter/zos/basics/index.jsp

Summary of changes

Summary of Changes
for SA23-2240-01
z/OS Version 1 Release 11

This document contains information previously presented in *z/OS JES Application Programming*, SA23-2240-00, which supports z/OS Version 1 Release 10.

New information

- Added a new section Special Processing for Logical SYSLOG Data Sets for Using the ACB/RPL Interface. See "Special Processing for Logical SYSLOG Data Sets" on page 10.
- Added a new section Return Codes for Using the ACB/RPL Interface. See "Return Codes" on page 11.

Changed information

- Updated the description for JES Spool Data Set Browse. See Chapter 2, "JES Spool Data Set Browse," on page 3.
- Updated sections Specifying the Data Set Name (DALDSNAM) and Building the Browse Token (DALBRTKN) for Allocation. See "Specifying the Data Set Name (DALDSNAM)" on page 4 and "Building the Browse Token (DALBRTKN)" on page 4.
- Added description to ACBCCANY in section Requesting Carriage Control. See "Using the ACB/RPL Interface" on page 8.

You may notice changes in the style and structure of some content in this document—for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

© Copyright IBM Corp. 2008, 2009

Chapter 1. Introduction

JES provides several application programs to supplement its performance. This document talks about procedures and considerations when you use the following application programs:

- Spool Data Set Browse
- Client/Server Print Interface
- The Internal Reader
- The External Writer

© Copyright IBM Corp. 2008, 2009

Chapter 2. JES Spool Data Set Browse

Spool data set browse (SDSB) is a function application program that can be invoked to process spool data sets. JES provides an interface that application programs can use for this purpose. SDSB can be used to access data sets that are still open in running address spaces. Any full data buffers that have been written to SPOOL can be read and optionally, data that has not been written (unwritten buffer support) can also be accessed.

Note: To read unwritten buffers with JES3, each of the following systems must be at z/OS V1R11, or higher:

- The system of the application.
- The system of the global processor.
- The system where the target data set is open.

If the application is running on the same system where the target data set is open, the system must be at z/OS V1R10, or higher.

Programs can use SDSB to allocate SPOOL data sets. The data set being requested is specified by passing the JES data set name along with a SPOOL browse token to MVS™ dynamic allocation. When the data set is allocated, the data set can be read using one of two methods:

1. Using the compatibility interface (DCB, GET).
2. Using the ACB/RPL interface.

You use the compatibility interface when synchronous sequential access is required. You use the ACB/RPL interface when random access to the spool file is required, or when asynchronous processing is required.

Allocation

JES recognizes an allocation of spool data set browse when a SPOOL browse token is specified on a dynamic allocation call. The SPOOL browse token is specified along with the data set name and contains optional control information JES uses to allocate the data set you want. SDSB can only be specified on dynamic allocation requests, and the following text units are required on the dynamic allocation request:

- DALDSNAM (data set name)

  Note: DALDSNAM can also be set from PDBDSNAM (JES2 only) or STVSDSN that is returned by extended status.
- DALSTATS (disposition = SHR)
- DALSSREQ or DALUASSR (subsystem name)

  Note: DALUASSR is the unauthorized version of DALSSREQ. They are mutually exclusive.
- DALBRTKN (browse token)

Other text units are optional. For example, DALRTDDN, used to return the ddname allocated to the data set.

Note: The DALSSREQ text unit can only be specified by authorized programs, and the allocation must be performed in authorized state. After the allocation is complete, an unauthorized task can perform I/O operations on the spool data set.

© Copyright IBM Corp. 2008, 2009

Specifying the Data Set Name (DALDSNAM)

The JES data set name is passed to dynamic allocation using the DALDSNAM key. The format of the data set name is:

```
userid.jobname.jobid.Ddskey.dsname
```

If the exact data set name is not known, the generic characters '?' and '*' can be used in the data set name. However the jobname and jobid are required. In the event that more than one data set matches the data set name requested, then the first data set that matches is allocated.

In addition to the standard JES data set name, there are some alternate data names that can be used to allocate specific JES data sets (without knowing the exact data set name) and logical data set concatenations. One alternate format is:

```
userid.jobname.jobid.jes_dsname
```

The *jes_dsname* is one of the following values:

- JCL - This is the input JCL (including the SYSIN data sets) exactly as submitted
- JESJCLIN - Same as JCL
- JESJCL - JCL images as output by the converter
- JESMSGLG - JES message log (WTOs issued by the job)
- JESYSMSG - JES system messages

Notes:

1. When JESMSGLG is used as the data set name, if JESLOG SPIN is specified, JES2 attempts to logically concatenate the spun off JESMSGLG data sets into a single logical data set.
2. When JESYSMSG is used as the data set name, if JESLOG SPIN is specified, JES2 attempts to logically concatenate the spun off JESYSMSG data sets into a single logical data set.
3. JES3 does not concatenate spun off JESMSGLG and JESYSMSG data sets into a single logical data set. The data sets are used as individual ones.

SPOOL Data Set Browse also supports accessing the current SYSLOG for a system (provided that there the system is maintaining a SYSLOG in this JESPLEX). To allocate the logical SYSLOG concatenation for a system specify the following data set name (in DALDSNAM):

```
sysname.SYSLOG.SYSTEM
```

The *sysname* is the MVS system name of the system whose SYSLOG the application will examine. JES logically concatenates all the SYSLOG data sets (even if there are multiple jobs) for the specified system in chronological order.

Building the Browse Token (DALBRTKN)

Before issuing the dynamic allocation request, the application must build the browse token and pass it with the DALBRTKN text unit. The format of the browse token is mapped by macro **IAZBTOKP**. The browse token is built in text unit format with 7 subparameters. When building the browse token, the following SVC 99 text unit fields must be set:

Field	Value
S99TUKEY	DALBRTKN
S99TUNUM	7
S99TUPAR	Mapped by IAZBTOKP

The token is of fixed length and all subparameters must be coded. Each browse token subparameter contains a length followed by the data, so it will be in text unit format.

You can complete the fields in the token as follows:

BTOKPL1
Length of the browse token identifier (LENGTH(BTOKID)).

BTOKID
Browse token id (BTOK). The IAZBTOKP macro defines constant BTOKCID to be used to set this field.

BTOKPL2
Length of the token version field (LENGTH(BTOKVER)).

BTOKVER
The 2 byte version number of the token parameter list. Byte 1 (or TOKTYPE) indicates the call type. If it is set to BTOKBRWS, then this is a normal browse request. If it is set to BTOKSTKN, then this is a SPOOL token based browse request. JES3 supports only BTOKSTKN. Byte 2 (or BTOKVERS) is the parm list version and should be set to BTOKVRNM.

BTOKPL3
Length of the IOT field LENGTH(BTOKIOTP).

BTOKIOTP
Data pointer whose content is based on the first byte of BTOKVER.

If BTOKVER is set to BTOKBRWS, this is the optional MTTR of the IOT containing the PDDB of the file to be allocated (obtained from JOEIOTTR or IOTTRACK, for example). JES3 does not support this form of BTOKIOTP.

If BTOKVER is set to BTOKSTKN, this must point to either a client token (returned from dynamic allocation using key DALRTCTK) or a data set token (returned by the SAPI SSI in field SSS2DSTR or the Extended Status SSI in field STVSCTKN).

BTOKPL4
Length of the job key field (LENGTH(BTOKJKEY)).

BTOKJKEY
Optional job key of the file to be allocated (for example, the job key obtained from JCTJBKEY, JQEJBKEY, or SJBJKEY). This field is not used if BTOKVER is set to BTOKSTKN. This field is required if BTOKTYPE is set to BTOKBRWS and BTOKIOTP is non-zero. JES3 does not support this parameter and this field is set to zero.

BTOKPL5
Length of the ASID field (LENGTH(BTOKASID)).

BTOKASID
The 2 byte ASID of the data set owning job if active buffers are needed. If active buffers are not needed, then pass 0. If the ASID is not known, then pass X'FFFF' and JES will determine the correct ASID.

BTOKPL6
Length of the RECVR field (LENGTH(BTOKRCID)).

BTOKRCID
Eight byte userid to be used as the RECVR on the SAF call or zeros if the

RECVR is not being used. JES uses this field to check authority to the browse request. When RECVR is used, the value must be left justified and padded with blanks.

When this parameter is specified, the *logstr* field should also be used so that usage of *recvr* can be logged. However, neither JES nor SAF enforces this convention.

BTOKPL7
Length of the *logstr* field (LENGTH(BTOKLOGS)).

BTOKLSDL
Length of the *logstr* (specified in field BTOKLSDA) to be used on the SAF call used by JES to check authority to the browse request, or zero if the *logstr* is not being used.

The *logstr* length must be a value from 0 to 254.

BTOKLSDA
Text of the *logstr* if BTOKLSDL is non-zero, or zeros if the *logstr* is not being used.

The maximum length text is 254 characters.

Note: When you use the compatibility interface to read the data set, you could also use text units specifying the record format, record length, and blocksize.

Security

JES does not perform any SAF call during allocation. When the SPOOL data is opened, JES uses SAF to verify read access to a JESSPOOL resource associated with the data set. SPOOL browse uses both the standard form of the JESSPOOL class resources and modified forms for special system data sets. Any generic characters that may have been specified at allocation are replaced by the actual values for the data set allocated.

When a logical data set name was specified for DALDSNAM, then the format of the resource name passed to SAF is:

```
localnodeid.userid.jobname.jobid.jes_dsname
```

In the resource name:

localnodeid
The NJE node name of the node on which the SYSIN or SYSOUT data set currently resides. The localnodeid appears in the JES job log of every job.

userid The userid associated with the job. This is the userid RACF® used for validation when the job runs.

jobname
The name that appears in the name field of the JOB statement.

jobid The job number JES assigned to the job. The jobid appears in notification messages and the JES job log of every job.

jes_dsname
One of the following fixed names:

- JCL - This represents the jobs input JCL (with all SYSIN data sets)
- JESJCL - The JCL images data set as created by the conversion process
- JESMSGLG - The JES2 job log data set
- JESYSMSG - The MVS SYSTEM messages data set

When a SYSLOG data set is allocated, the format of the resource name passed to SAF is:

```
localnodeid.+MASTER+.SYSLOG.SYSTEM.sysname
```

In the resource name:

localnodeid
The NJE node name of the node on which the SYSLOG data set resides. The localnodeid appears in the JES2 job log of every job

sysname
The MVS system name of the system that created the SYSLOG.

If the browse token specifies a *recvr* userid, the SAF call is performed with the RECVR parameter. When the *recvr* userid is specified, the *logstr* parameter should also be supplied.

If the data set fails the security check, the open request fails with R15=0C and an error code stored in ACBERFLG (decimal 152).

The system performs a SAF call as part of OPEN processing to ensure that the user is authorized to the data set. In JES2, if the user is not authorized, a system abend, code S913, results. In JES3, although control is returned to the application, the DCBOFOPN bit is not set and the application cannot read the data set. After the DCB has been opened, use a GET macro pointing to the DCB to read the file. When processing is complete, use a CLOSE macro to close the file. The same task that opened the DCB must be used to close it.

Errors and Return Codes

Three types of errors might occur:

- If an error occurs allocating the data set, in both JES2 and JES3, dynamic allocation returns an S99ERROR reason code of X'04F8' describing the error. You can use the DAIRFAIL service to format the error text.
- If, during allocation, JES2 detects an un-initialized data set (that is, PDBMTTR contains zeros), it fails the dynamic allocation also with an S99ERROR reason code of X'04F8'. This condition does not apply to JES3.
- If, during data set open, an error occurs:
  - JES2 sets a return code in register 15 following the OPEN and stores a reason code in the ACB.
  - JES3 does not set upon the flag ACBOPEN or DCBOFOPN returned from the OPEN macro.

Using the Compatibility Interface

After allocating the spool data set, you can use the compatibility interface to sequentially read each record. Your application program builds and opens a DCB that specifies the ddname of the allocated spool data set. The record format, record length, and block size could be specified on the allocation or in the DCB, or they could be obtained from the SYSOUT data set that was allocated.

Using the ACB/RPL Interface

Your application can use the ACB/RPL interface to obtain the most flexibility when using spool data set browse (SDSB). With this method, the application builds and opens an ACB (access method control block), and uses RPL based macros to read and position the data set. You use a subset of the ACB/RPL macros as documented for VSAM. In general, JES implements only those features required to process the data set. Other options specified on the macros are ignored. When coding the ACB and RPL macros, AM=VSAM should be specified or defaulted.

Building the ACB

After the dynamic allocation completes for the data set, your application builds and opens an ACB. You can use the GENCB service or map the ACB with IFGACB macro. The storage for the ACB must be resident below the 16 megabyte line. You then use the SHOWCB and MODCB services to display and modify selected fields of the ACB. However, some of the fields required by JES are not processed by those services, and hence they may be of limited use. The only required field for the ACB is the ddname to use. The ddname can be assigned when the spool data set is allocated, or the system can return a generated name. In either case, the ACBDDNAM must be completed before the ACB is opened.

Note: The ACB can also specify an optional exit list. The exit list is built with the EXLST macro, and can be used to specify the EODAD, SYNAD, and LERAD exits.

When the ACB is generated, your application opens it using an OPEN macro. As part of open, the system performs a SAF call to ensure that the user is authorized to the data set. In JES2, if the user is not authorized, a system abend, code S913, results. In JES3, although control is returned to the application, the ACBOPEN bit is not set and the application cannot read the data set.

Note: Unless authorization has changed between the time the data set was allocated and opened, an unauthorized user will be detected at allocation time.

After the open is successful, you use RPL based macros to read the data set.If the open is not successful, error and reason codes that describe the error are placed in the ACB.

When processing is complete, your application should issue a CLOSE macro specifying the open ACB. If a CLOSE is not performed, an automatic close occurs by the system when end-of- task occurs. The close must be done by the same task that opened the ACB.

Requesting Carriage Control

The spool data set browse (SDSB) interface can be used to obtain the carriage control character if it is contained in the record. The spool data set browse interface returns carriage control with the record only if it is requested. Your application indicates that carriage control is wanted by setting flag ACBCCTYP as follows:

- ACBCCTYP.ACBCCASA = ON indicates that ASA characters are wanted.
- ACBCCTYP.ACBCCMCH = ON indicates that machine characters are wanted.

Both bits can be turned on to indicate that carriage control is required regardless of format. When a data set contains carriage control, and the application requests carriage control for that type, the control character will be returned in the first byte of the record area returned by JES.

If the data set does not contain carriage control, but the application requests carriage through the setting of ACBCCTYP, JES performs the following:

- If ACBCCASA is on, JES returns a X'40' as the carriage control.
- If ACBCCMCH is on, JES returns a X'09' as the carriage control.
- If both ACBCCASA and ACBCCMCH are on, JES2 returns a X'09' and JES3 returns a X'40' as the carriage control.

To obtain the exact carriage control associated with a record and an indicator of the carriage control type, an application should set the ACBCCANY bit. If this bit is set, JES returns a pointer to the carriage control in RPLCCHAR and the type of carriage control in RPLOPT4.

Using RPL Based Macros

All I/O requests are specified using an RPL. The RPL contains the address of an open ACB. You build the RPL using the GENCB service, or map it with the IFGRPL macro. You can then use SHOWCB and MODCB for some functions. The storage for the RPL must be resident below the 16 megabyte line.

The processing options are specified in the RPLOPTCD parameter. Only a subset of those defined by VSAM are recognized by JES. In particular, spool data set browse supports only ″move mode″ (OPTCD=MVE) processing. OPTCD=SYN can be specified for synchronous requests (the default), or OPTCD=ASY can be used for asynchronous processing.

Your application must obtain a buffer area where JES places the record it reads. The address of the area is placed in RPLAREA and its length in RPLBUFL. If the area is not large enough to contain the record, JES sets an error code in the RPL. The storage for the buffer may be resident above the 16 megabyte line.

Your application reads each record using a GET macro which points to the RPL. When synchronous processing is used, control returns to the application when the GET is complete and the record has been moved to the application provided area. JES sets the length of the returned record in field RPLRLEN.

When asynchronous processing is used, your application must issue a CHECK macro specifying the RPL that is to be waited on. Control then returns to your application when the function specified by the RPL is complete.

After each GET request, JES returns a token your application can use to position directly to the record if it needs to be reread. JES returns the 8 byte token in field RPLRBAR after each GET. Your application should treat RPLRBAR strictly as a token, and not depend on it to be in a specific format.

Your application can use the POINT macro to locate a previously read record. POINT specifies an RPL that contains an RPLRBAR that was returned on the GET request for the record you want. POINT positions JES to the record to be read; it does not read the record. A GET macro must be issued after the POINT to retrieve the record. Input to POINT is an 8 byte RBA value pointed to by RPLARG. This RBA value can be either a RPLRBAR value saved from a previous GET or a record number relative to the start of the data set (0-16,777,215). If the record requested is

not found, POINT processing will fail, leaving the data set in an undefined state. A GET request after a failed POINT will result in unpredictable data being returned.

Each GET request returns a logical record in the file. The application is isolated from the internal format of the record. JES places the complete record in the RPLAREA buffer and its length in the RPLRLEN field. JES performs the necessary unwritten buffer support for active jobs; however, no indication of the source of the record is placed in the RPL. JES places feedback information in the RPL after each request. RPLRTNCD and RPLCNDCD should be checked for satisfactory completion after each operation.

Special Processing for Logical SYSLOG Data Sets

When a SYSLOG data set is allocated by using the special data set name of sysname.SYSLOG.SYSTEM, a number of additional options are available. This is in addition to all the options available to normal SPOOL Data Set Browse processing. Additional information can be returned on a successful GET request. To obtain this information, do the following settings:

- Set RPLERMSA to the address of a data area mapped by IAZDSINF.
- Set RPLEMLEN to the length of the data area (DSINSIZ1).
- Set DSINEYE eyecatcher to the value of DSIN.

The following information can be returned:

- The source record number (relative to this data set and the beginning of the concatenation).
- The time stamp associated with the message (STCKE format).
- The source job number and data set number for the record.

In addition, POINT processing is enhanced to provide support for additional RBA formats. All new RBA formats start with a X'FF', which is followed by a 1-byte function number and a 6-byte argument. Table 1 shows the RBA values (in hex) that JES supports.

Table 1. Additional RBA formats

RBA value (in hex)	Function
FF00*cccc cccccccc*	Go to first occurrence of time stamp passed in *cccc cccccccc* (STCKE format)
FF01*cccc cccccccc*	Go to next record with a time stamp of *cccc cccccccc*
FF02*cccc cccccccc*	Go to previous record with a time stamp of *cccc cccccccc*
FF03*rrrr rrrrrrrr*	Move *rrrr rrrrrrrr* records (signed value) from the current record
FF04*rrrr rrrrrrrr*	Move to absolute record *rrrr rrrrrrrr* from start of the (logical) data set

Note: Messages in the log might not be in chronological order. This must be considered when search for records based on a time stamp.

POINT processing that specifies a specific time (RBA values starting with X'FF00'), an absolute record number (RBA values starting with X'FF04'), or a relative record

number (RBA values starting with X'FF03') will position the data set to the first or last record of a file if the argument specified if beyond the end or before the start of the data set.

Return Codes

By using the ACB/RPL interface, the requests of JES I/O return error information in a 3–byte RPL field, RPLFDBK. The list below explains the meaning of the these bytes:

Byte **Meaning**

RPLRTNCD Requests return code.

RPLNOERR - X'00'
No error encountered.

RPLLOGER - X'08'
Logic error encountered.

RPLPHYER - X'0C'
Physical error encountered.

RPLCMPON Component processing request.

X'02'
JES2 processing request.

X'03'
JES3 processing request.

RPLERRCD Error code associated with return code. The meaning of the error code is based on the request return code:

- For request return code RPLLOGER, RPLERRCD could be one of the following values:

  RPLEODER - X'04'
  Normal end of data has occurred.

  RPLNOREC - X'10'
  A POINT request was issued but the record specified by the RBA passed in RPLARG was not found. Another POINT must be done before any further GETs.

  RPLINRBA - X'20'
  An RBA associated with a POINT or a GET-UPDATE request was found to be invalid (not a recognizable format).

  RPLNOVRT - X'28'
  The request required the scheduling of an SRB to complete but the needed virtual storage could not be obtained.

  RPLINBUF - X'2C'
  On a GET request, the size of the area (RPLBUFL) passed in RPLAREA was too small to contain the record being returned. The actual record size is set in RPLRLEN. Obtain a larger area and re-issue the GET request.

  RPLINACC- X'44'
  Access type is not allowed for this data set. For example, an attempt to PUT to a data set that was opened for input processing.

RPLINUPD - X'5C'
PUT-UPDATE occurred before GET-UPDATE.

RPLDLCER - X'64'
PUT-UPDATE length was changed.

RPLINLEN - X'6C'
A PUT was done for a record and the specified length exceeds the JES limit of 32767 bytes.

RPLNOBFR - X'98'
A POINT request was made but all available buffers are being used by outstanding locate mode GET requests.

RPLREOB - X'20'
A GET request was made but all available buffers are being used by outstanding locate mode GET requests.

- For request return code RPLPHYER, RPLERRCD could be one of the following values:

RPLRDERD - X'04'
The read request failed because either there was an physical read error or the record read did not pass validation processing.

RPLWTERD - X'10'
A write operation encountered a physical write error.

End of File Processing

For jobs not running, the spool data set browse (SDSB) interface utilizes the end-of- file exit when no more data can be read. When using the compatibility interface, the exit is defined using the EODAD parameter of the DCB. When using the ACB/RPL interface, the end-of-file exit is defined on the EXLST macro. In addition, the return codes are placed in the RPL to indicate the end of file condition.

For jobs actively running, using the ACB/RPL interface, your application attempts to read unwritten spool buffers that reside in the address space of the active job. When all unwritten buffers have been read, the ACB/RPL utilizes the end-of-file exit. Your application can issue subsequent get requests to obtain additional data. On each get request, the ACB/RPL interface returns the unwritten buffer data, if available. Thus, an end of file condition for an active job should be considered temporary rather than permanent. As the active job creates additional data, subsequent get requests can be used to retrieve it. If no more data is available at the time of the get, end of file will be driven.

Note: This processing differs from standard access methods. Normally, a get request issued after end of file is considered as a permanent error. However, this condition should be expected when using the spool data set browse interface against an active job.

Performance

Spool data set browse (SDSB) presents a record level interface (that is, a complete record is returned on each get request). If a record is internally stored in several JES spool blocks, your application's use of the interface performs the necessary spool I/O to assemble the record segments. JES maintains only one spool buffer in storage at a time. Similarly, if a POINT macro is issued for a record not contained in the last spool block, JES initiates the I/O to read the block.

Your application can improve the performance of the interface by optimizing the number of I/O requests. For example, you might want your application to buffer previously read records in storage, rather than call the interface with POINT and GET to reread a record.

Secondary Subsystem Support

The spool data set browse (SDSB) interface supports allocation of spool data sets to secondary JES2 subsystems. Specify the subsystem name in the DALSSREQ or DALUASSR text unit and direct the allocation to the proper JES2.

Notes:

1. JES3 cannot be a secondary subsystem.
2. A secondary subsystem cannot be halted until all outstanding allocations are freed.

Chapter 3. JES Client/Server Print Interface

JES provides an interface for a job to function as a server and make SYSOUT requests on behalf of a client.

There are several ways in which a data set created by a server differs from a data set created by an ordinary SYSOUT DD or dynamic allocation.

1. Data sets created by a server use the DALRTCTK dynamic allocation text unit, which causes JES to create a unique **Client Token**(CTOKEN) associated with the data set from that point on.
2. The server can use the Extended Status or SYSOUT Application Programming Interface (SAPI) Subsystem Interface (SSI) calls to access a data set, specifying a CTOKEN in the selection criteria in order to request a particular data set, without needing to know any other information about the data set.
3. When a data set has a CTOKEN, JES informs the application, through the use of ENF signal 58, of events relating to the data set. Among these events are selection by a writer, deselection by a writer, and data set purge. JES also issues signals for important events related to any job that has created at least one data set with a CTOKEN, such as job purge.

Creating a CTOKEN

The server creates an 80-byte CTOKEN using the Dynamic Allocation text unit of DALRTCTK. The DALRTCTK text unit appears as follows:

+0	**+2**	**+4**	**+6**
DALRTCTK	00 01	00 50	Returned data

Upon return from SVC 99, the data starting at position 6 in the CTOKEN text unit contains the CTOKEN returned by JES, provided the allocation was successful. When a CTOKEN is returned to you, add it to your list of CTOKENs for later use.

CTOKENs contain internal information that JES uses to locate the data set when the server issues SSI requests. Once you have received a CTOKEN from JES, do not change its contents except in one special case that will be discussed later.

CTOKENs contain ordering information. This allows you to store CTOKENs in a data structure of your choice that can make use of the order and result in faster searches. CTOKENs are not ordered according to a creation timestamp; they are ordered internally by JES.

Refer to the topic about SSI Function Codes Your Program Can Request and SSI 54 in *z/OS MVS Using the Subsystem Interface* for a detailed description of the Subsystem Version Information Call.

Comparing CTOKENs

At various times during your processing, you will need to compare CTOKENs. Typically, you will do this when JES signals an event for a CTOKEN and you need to find this token in your list so that you can take some kind of action based on the event.

© Copyright IBM Corp. 2008, 2009

To compare one CTOKEN to another, you must not simply compare the entire 80 byte values. This is because under certain JES processing, CTOKEN equality is based on a subset of the information in the CTOKENs matching while other information in the CTOKENs could be different. IBM provides a macro IAZXCTKN which you must use to compare CTOKENs. This macro determines which information in two CTOKENs is significant and compares just this information. The macro works in such a way that you never need to interpret any information inside the CTOKEN.

Depending on the return code from the IAZXCTKN macro, you can determine whether the two CTOKENs are the same, whether the first CTOKEN is less than the second one, or whether the second CTOKEN is less than the first one.

IAZXCTKN also provides a special comparison function. At certain times, JES signals events for an entire job. When this happens, the signal includes a **job level CTOKEN**. Using IAZXCTKN, you can determine whether a CTOKEN for a data set in which you are interested is covered by the job level CTOKEN that JES provides.

Job level CTOKENs contain no ordering information; therefore a job level CTOKEN can be considered by IAZXCTKN to be ″equal″ or ″not equal″ to another CTOKEN but never ″greater″ or ″less″ than another CTOKEN.

Refer to the book *z/OS MVS Programming: Authorized Assembler Services Guide* for information about using the IAZXCTKN macro.

Obtaining Status for a Data Set

You can obtain status for a data set using the Extended Status subsystem interface (SSI 80) code. To do this, you supply as STATCTKN the address of the CTOKEN for the data set you are interested in and set the selection flag STATSCTK. When you use the STATSCTK selection flag, you cannot use the STATSJBI selection flag, and vice versa.

Refer to the topic about SSI Function Codes Your Program Can Request and SSI 80 in *z/OS MVS Using the Subsystem Interface* for a detailed description of the Extended Status call.

Accessing a Data Set

You can access a data set using the SYSOUT Application Programming Interface, SSI 79. You would do this in order to:

- Show the contents of the data set to the requesting client.
- Allow the client to delete a data set.
- Allow the client to release a data set from hold to print.

To request JES to perform a SAPI operation on a client data set, you supply as SSS2CTKN the address of the CTOKEN for the data set you are interested in and set the selection flag SSS2SCTK. When you use the SSS2SCTK selection flag, you cannot use the SSS2SJBI selection flag, and vice versa.

When a data set is processed by a program written using SAPI, there is a distinction between a data set that is selected for processing and a data set that is selected for browsing. In the former case, the intention is to select the data set in much the same way as it would be selected for a writer (such as an external writer), which may or may not cause its state to be changed. In the latter case, the

intention is to not change its state at all. The main purpose for this distinction is to prevent ″noise″ caused by unnecessary ENF signals.

You can set the flag SSS2SBRO when you know that the intention of a SAPI access is to browse a data set. When this flag is on, JES will not issue any signals for the SAPI access to this data set. When this flag is off JES will issue signals whenever a data set with a CTOKEN is selected or deselected by a SAPI Put/Get operation. Do not set this flag if you need to be informed of selects and deselects.

This flag controls signals for selects and deselects only. If a data set is purged by a SAPI Put operation (for example, by turning off flag SSS2DKPE), a signal will be issued even if SSS2SBRO is set.

You must use SAPI in order to suppress signals when accessing a data set for browse. When a Process Sysout (PSO) application selects or deselects a data set with a CTOKEN, a signal is always issued.

The SSS2SBRO flag is valid only for Put/Get requests.

Refer to the topic about SSI Function Codes Your Program Can Request and SSI79 in *z/OS MVS Using the Subsystem Interface* for a detailed description of SAPI.

Security

Since all SYSOUT allocations and SAPI calls are being done by you as the server, preventing a client from having unauthorized access to another client's data set is your responsibility.

One way you can do this is by performing the dynamic allocation to create the SYSOUT file under a security environment with the client's identity. This is accomplished by using the RACROUTE macro with REQUEST=VERIFY. Then, when a client makes a request requiring you to make a SAPI SSI call, you would use RACROUTE REQUEST=VERIFY with the requesting client's user id to establish a security environment for the requestor. As part of the SAPI processing, JES makes authorization checks using the JESSPOOL security class.

Refer to the book *z/OS Security Server RACROUTE Macro Reference* for information on using the RACROUTE macro.

This method requires your clients to be defined as users in your security product, even if they never directly log on to your system. If this is not possible, you must design your own security protocol.

Identifying a Requestor on a Header Page

JES typically has printers defined to print with a header page identifying the job creating a SYSOUT data set.

However, the job information that prints on the header page is associated with the job that created the data set. This ordinarily identifies the job that runs your server, not the client that requested the printout. You would probably prefer that the client's identification print rather than the server's in order to be able to tell one client's output apart from another's.

In order to do this, you can use the IAZXJSAB macro. You could do something like the following:

```
IAZXJSAB CREATE,JOBNAME=client_jobname,
          USERID=client_userid,TYPE=SUBTASK
```

The CREATE call must be made prior to allocating the dataset in both the DYNALLOC and ALLOC cases.

To make sure that the job identification does not persist beyond the requested data set, you can delete the JSAB after the first OPEN for the dataset by making the following call:

```
IAZXJSAB DELETE,TYPE=SUBTASK
```

or you can update it with different user identification after the first OPEN by making the following call:

```
IAZXJSAB UPDATE,JOBNAME=new_client_jobname,
          USERID=new_client_userid
```

In the CREATE and DELETE cases, you must use the parameter TYPE=SUBTASK, otherwise JES will not recognize the requesting user identification correctly.

See the topic about the IAZXJSAB macro in *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG* for additional information about using the IAZXJSAB macro.

Listening for Events

During the course of JES operations, data sets and jobs are subject to changes for various reasons. When such events occur for a client data set (for example, a data set that was allocated with the DALRTCTK text unit) or a job containing at least one client data set, JES issues an Event Notification (ENF) signal. The ENF number of the signal is 58. The signal is issued only for data sets that have been allocated using the DALRTCTK text unit.

To listen for this signal, you could do something like the following:

```
ENFREQ ACTION=LISTEN,CODE=58,EXIT=exit_address,XSYS=YES,
     PARM=parameter_address,DTOKEN=end_token_address
```

Note: The XSYS=YES parameter is used because JES could be issuing signals on a different processor from the one where your server runs.

To stop listening for this signal, you could do something like the following:

```
ENFREQ ACTION=DELETE,CODE=58,DTOKEN=end_token_address
```

The data area received by your listen exit from ENF is mapped by the IAZENF58 macro. You must include this macro in your program in order to use the data supplied by the ENF signal. This data area contains the following information:

ENF58_LENGTH
Length of parameter list

ENF58_QUALIFIER
Qualifier code — defined below:

ENF58_Q_PURGE
Data set was purged

ENF58_Q_SELECT
Data set was selected

ENF58_Q_DESELECT_PROCESSED
Data set was processed

ENF58_Q_DESELECT_NOT_PROCESSED
Data set is no longer selected, disposition was not changed

ENF58_Q_DESELECT_NOT_PROCESSED_HELD
Data set is no longer selected, disposition was not changed and data set is held

ENF58_Q_DESELECT_ERROR
An error resulting in a system level hold occurred

ENF58_Q_EOD_OK
End of data set notification occurred — successful

ENF58_Q_EOD_ERROR
End of data set notification occurred — unsuccessful

ENF58_Q_JOB_CHANGE
Job-status change occurred

ENF58_Q_TOKEN_CHANGE
Client token has changed

ENF58_Q_CHECKPOINT
A checkpoint has occurred on the printer on which the data set is printing.

ENF58_SYS_HOLD
System hold reason — refer to IAZOHLD for possible values

ENF58_JES_NAME
JES2 Member Name / JES3 MAIN name

ENF58_REASON
Reason text

ENF58_CTOKEN
Data Set Client Token

ENF58_NEW_CTOKEN
New client token that should replace the CTOKEN for a TOKEN_CHANGE ENF type

You should determine what action you need to take based on this event. For example, if you receive a signal with ENF58_Q_PURGE it usually means that you should delete from your list all information pertaining to the dataset with the CTOKEN of ENF58_CTOKEN.

To take action on the CTOKEN, you must first go through your CTOKEN list and issue IAZXCTKN macros, comparing ENF58_CTOKEN to CTOKENs from your list until you find the CTOKEN specified in the signal in your list. If ENF58_QUALIFIER is ENF58_Q_JOB_CHANGE, it means that ENF58_CTOKEN is a job level CTOKEN and you must go through your entire list of CTOKENs until you have identified, and taken action on, all data set level tokens covered by the job level CTOKEN.

Notes:

1. When ENF58_QUALIFIER is ENF58_Q_JOB_CHANGE, the CTOKEN in ENF58_CTOKEN is a job level CTOKEN. At all other times it is a data set level CTOKEN.

2. When ENF58_QUALIFIER is ENF58_Q_TOKEN_CHANGE, the ENF58 parameter list contains a new CTOKEN and ENF58_LENGTH reflects the existence of this new CTOKEN.
3. When an event with ENF58_Q_TOKEN_CHANGE is received, the CTOKEN in your list should be replaced with the contents of ENF58_NEW_CTOKEN. This is the only time that you should change the contents of a CTOKEN. Replacing this CTOKEN does not change the ordering of the CTOKEN you previously had in your list for this data set.
4. ENF58_NEW_CTOKEN is present only when ENF58 QUALIFIER is ENF58_Q_TOKEN_CHANGE. ENF58_LENGTH is larger for this qualifier type than it is for other types.
5. When an event with ENF58_Q_CHECKPOINT is received, the below fields are also present and contain the status of the print at the time that the checkpoint is taken.

   ENF58_COPY
   Checkpointed copy count

   ENF58_RECORD
   Checkpointed current record

   ENF58_PAGE
   Checkpointed current page

   These fields are present only when ENF58_QUALIFIER is ENF58_Q_CHECKPOINT. ENF58_LENGTH is larger for this qualifier type than it is for other types except ENF58_NEW_CTOKEN.

   If checkpoints occur frequently, ENF58 checkpoint signals may be generated at a fast rate.

   If a restart of JES or the printer occurs after a checkpoint, the next checkpoint could be for a page or record count that represents reprocessed records or pages. This is because after a restart a writer will continue at the last checkpointed record or page, not the last one that completed printing.

See *z/OS MVS Programming: Authorized Assembler Services Guide* and *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG* for information about using the ENFREQ macro and coding the listen exit.

Chapter 4. Internal Reader Facility

The internal reader facility is a logical device similar to a card reader that allows you to submit jobs to JES. You can also read job streams from tape, disk or any QSAM-supported device through the internal reader to JES by using the procedure named RDR. Using the internal reader facility, you can submit jobs from time-sharing logons, started tasks, or other jobs. The ability to submit jobs from currently running jobs or tasks is especially powerful. This ability gives the programmer the flexibility to have a job that reaches a point successfully to submit another job for execution.

Defining the internal reader facility

In JES2, define the attributes of the internal reader facility with the INTRDR statement.

There are three types of internal readers:

- TSO logons are submitted by use of TSOINRDR. TSUINRDR and TSOINRDR are used interchangeably.
- Started tasks are submitted by use of STCINRDR.
- Batch jobs are submitted by use of INTRDR.

In JES3, internal readers are dynamically managed by the JES3 global and are always available for use.

If BATCH=NO is specified, you cannot use internal readers for batch jobs. However, you can still submit batch jobs through real (local) card readers, RJE, NJE, or spool offload.

Using the internal reader facility

There are four methods of using the internal reader facility. These methods are:

- Using a special external writer called INTRDR to submit a job from input in a batch job stream.
- Dynamically allocating the internal reader from your program.
- Using the IBM-supplied RDR procedure from either a batch job stream or the operator's console to read the job from a QSAM-supported device.
- Using the TSO/E SUBMIT command to pass a job stream to the internal reader facility. For more details about the TSO/E SUBMIT command, see *z/OS TSO/E Command Reference*.

Submitting to the internal reader from jobs or tasks

Figure 1 on page 22 shows a step from a job (or task) that submits a job to the internal reader.

© Copyright IBM Corp. 2008, 2009

```
⋮
//STEP9     EXEC  PGM=IEBGENER
//SYSPRINT  DD    SYSOUT=Z
//SYSUT2    DD    SYSOUT=(A,INTRDR)
//SYSIN     DD    DUMMY
//SYSUT1    DD    DATA,DLM=XX
//MYJOB1    JOB   ACCT,VAZQUEZ,CLASS=A
//STEP1     EXEC  PGM=CRUSHER
//ERRORS    DD    SYSOUT=A
//INPUT     DD    DSN=JES2.INIT.TUNE,DISP=SHR
//OUTPUT    DD    DSN=SMALL.BOOK,DISP=SHR
XX
//STEP10    EXEC  ...
⋮
```

Figure 1. Submitting a Job to the Internal Reader

Step 9 writes the JCL that follows the STEP9 SYSUT1 card (up to the XX which acts as a delimiter) to a SYSOUT data set used as input to the INTRDR program.

If the ACB interface was used to open the internal reader, you can use the ENDREQ macro to complete the submission of jobs. For more information about coding the ENDREQ macro, see *z/OS DFSMS Macro Instructions for Data Sets* and *z/OS Communications Server: SNA Programming*. For more information about JES control statement processing, see “JES control statements that affect the internal reader” on page 24.

Dynamically allocating the internal reader

You can allocate SYSOUT data sets to the special external writer, INTRDR, just as you would any other external writer. For example, your program can issue an SVC 99 (for details on SVC 99, see *z/OS MVS Programming: Assembler Services Guide*) and write JCL-images directly to the internal reader.

When you dynamically allocate the internal reader, the default MSGCLASS for the submitted job is the same as the MSGCLASS of the job or TSO/E logon that submits the job.

Time-sharing logon (TSO/E) and started task (STC) flow

Time-sharing logons and started system tasks appear to JES as two special forms of jobs that are received from designated internal readers. In JES2, these jobs are queued in special job classes (TSU and STC) and are assigned a MSGCLASS that is set during JES2 initialization (MSGCLASS parameter on the JOBCLASS(TSU) and JOBCLASS(STC) initialization statement). In JES3, the MSGCLASS for these jobs defaults to the MSGCLASS parameter specified on the CIPARM initialization statement. The two byte of PARMID= parameter on CIPARM statement is referenced by the INTPMID= parameter on the STANDARDS initialization statement.

The time-sharing message class (MSGCLASS parameter on the JOBCLASS(TSU) or CIPARM statement) becomes the output class for all dynamically allocated SYSOUT data sets for which a class is not specified, and becomes the MSGCLASS for all submitted jobs with no MSGCLASS parameter on the JOB statement. It is, therefore, not advisable to set MSGCLASS= to a SYSOUT class that specifies OUTDISP=PURGE. See the topic about Output disposition for SYSOUT data sets in *z/OS JES2 Initialization and Tuning Guide* for further information.

Time-sharing users can dynamically allocate data sets, dynamically deallocate them (spinoff), and print them at the time-sharing terminal (OUTPUT command). JES treats a file submitted by the TSO/Extensions interactive data transmission facility as an output data set.

Using the RDR procedure

Figure 2 shows the JCL procedure IBM supplies for you to use the internal reader to read jobs from tape, disk, or any QSAM-supported device.

```
//IEFPROC     EXEC   PGM=IEBEDIT
//SYSUT1      DD     DDNAME=IEFRDER
//IEFRDER     DD     DSN=NULLFILE,DISP=OLD
//SYSUT2      DD     SYSOUT=(A,INTRDR)
//SYSPRINT    DD     SYSOUT=A
//SYSIN       DD     DUMMY
```

Figure 2. The RDR Procedure

Examples of using the RDR procedure

The operator can invoke the RDR procedure to read:

- A job stream from the second file of a tape named JOBTAP on device 180:

  ```
  S  RDR,180,JOBTAP,LABEL=2,DSN=JOBS
  ```

- A job stream from a cataloged library of jobs:

  ```
  S  RDR,3330,DSN=PRODUCTN(PAYROLL)
  ```

- A job stream starting with a specific job on a tape named JOBTAP, the operator must submit a job to JES2 similar to:

```
//READJOBx    JOB    ...
//            EXEC   RDR
//IEFRDER     DD     DSN=JOBS,VOL=SER=JOBTAP,
//                   UNIT=3400,DISP=OLD
//SYSIN       DD     *
              EDIT   START=JOBx
/*
```

By using conditional JCL, you can cause internal readers to start only under specific conditions. You can then form a dependent job or set of jobs that execute (without operator intervention) only when a master job executes in a manner you want.

For example, to submit BADNEWS only if GOODNEWS does not complete successfully, specify the following:

```
//STEPTHEN IF (RC = 0)THEN
//*
//GOODNEWS EXEC PGM=IEBGENER
//SYSPRINT DD DUMMY
//SYSIN  DD DUMMY
//SYSUT1  DD JOBS(JOBA)
//SYSUT2  DD SYSOUT=(A,INTRDR)
//*
//STEPELSE ELSE
//BADNEWS EXEC PGM=IEBGENER
//SYSPRINT DD DUMMY
//SYSIN  DD DUMMY
//SYSUT1  DD JOBS(JOBB)
//SYSUT2  DD SYSOUT=(A,INTRDR)
//*
//STEPEND ENDIF
```

User-written procedures and programs can further exploit the internal reader facility to select particular jobs, to generate special job streams, and to allow operator submission of production job streams.

JES control statements that affect the internal reader

The following JES control statements affect the way in which the internal reader handles the input stream it receives:

- /*EOF - ends the current job in the data set and makes it eligible for immediate processing.
- /*DEL - deletes the job in the data set and schedules it for immediate SYSOUT processing. This statement deletes the current job in the job stream. If there is no job in the data set, this statement has no effect. The SYSOUT consists of any JCL submitted, followed by a message indicating that the job was deleted before execution.
- /*SCAN - causes the job to be scanned for JCL errors, but not executed. (The same processing occurs if TYPRUN=SCAN appears on the JOB statement.)
- /*PURGE - deletes the job in the data set and schedules it for purge processing. If no job is in the data set, this statement deletes the previous job in the job stream. No output is produced for this job. This is for JES2 only because JES3 does not recognize /*PURGE as a control statement.

Performance considerations for JES internal reader

The following performance considerations affect the performance of internal reader in a JES subsystem.

Use of unblocked records for SYSIN and SYSOUT data sets

You should not block SYSIN and SYSOUT data sets because the SAM (sequential access method) compatibility interface will increase overhead by unnecessarily deblocking and blocking data sets.

Held internal readers in JES2

JES2 treats all internal readers as a single facility, therefore holding one internal reader places all internal readers in hold. This is particularly troublesome when the central operator holds the internal readers and TSO/E users want to submit jobs.You can avoid this problem by:

1. Assigning a specific job class for all jobs submitted through a particular internal reader. Instead of holding the internal reader, you can hold the class by using either a JES2 initialization statement or a JES2 $T JOBCLASS(x),QHELD=YES command.
2. Use the TYPRUN=HOLD parameter or TYPRUN=JCLHOLD parameter on the JOB statement.
3. Submitting the job through an internal reader and individually hold it with the JES2 $H J command.

Record length of SYSIN data sets

Jobs can include input data in SYSIN data sets. In JES2, the maximum length of a record written to the internal reader is 32760 bytes. In JES3, the maximum length is the installation defined buffer size. These can be processed locally or sent to other nodes through NJE. Some NJE nodes do not support SYSIN records that are greater than 254 bytes (in JES2) or 80 characters (in JES3) in length. When data is sent to one of these nodes, the SYSIN records will be truncated to 254 bytes (in

JES2) or 80 characters (in JES3). Before attempting to send long SYSIN records to a node, ensure that the node and any intermediate node support long SYSIN records (for example, by sending a test).

SYSIN record formats

JES sets the record format for SYSIN data sets based on the data written to them. If all records that are written are of the same length (before any blank truncation), the record format (RECFM) will be set to fixed (F). If the records vary in length, the record format will be set to V. If carriage control is detected in the SYSIN stream, the record format will be updated to FM, FA, VB or VA depending on whether the records vary in length or not and whether the carriage control is ASA or Machine. If both ASA and Machine carriage control are detected, the record format will be set in the RECFM.

Chapter 5. The External Writer

This chapter documents Intended Programming Interface and Associated Guidance Information.

An external writer allows an installation to perform output processing for data sets not eligible for processing by the primary job entry subsystem (JES2 or JES3). For example, an external writer processes data sets going to printers, plotters, diskettes or data sets that are to be stored on a device not supported by JES.

There are two major parts of the IBM-supplied external writer that can be modified or replaced by an installation: the output writing routine and the output separator routine.

This chapter describes:
- Overview of the IBM-supplied external writer
- How to set up and start the IBM-supplied external writer
- How the IBM-supplied output writing routine works
- How to add your own output writing routine
- How the IBM-supplied output separator routine works
- How to add your own output separator routine

See *z/OS MVS Using the Functional Subsystem Interface* for information in subsystem interface (SSI) function code 1 on how to write your own external writer.

Overview of the IBM-Supplied External Writer

IBM supplies an external writer, which contains two routines that can be modified or replaced by the installation:

- The output writing routine. This routine processes data sets.
- The output separator routine. This routine writes separator records within a continuous listing or deck to help the operator separate one job's output from another's.

Characteristics of the IBM-Supplied External Writer

The external writer has the following features:

- It runs as a started task, in its own address space, in 24-bit addressing mode.
- It processes only those data sets that meet its selection criteria. You set some of these in a cataloged procedure (see "The External Writer Cataloged Procedure" on page 28) but you can override them with the START and/or the MODIFY operator commands (see *z/OS MVS System Commands* for the general description of these commands).

  If you do not specify data set selection criteria, the external writer will be used to process **all** the output in the system.
- It removes data sets from the JES spool. That is, it dynamically allocates the data sets, reads them, writes them to output devices, and then dynamically deallocates them. A large format spool data set can be shared.
- It supports basic format, extended format and large format sequential data sets on DASD.
- It supports checkpoint and spool on 3390 Model 9 and Enterprise Storage Server® (ESS).
- It accesses data sets according to one or more of the following criteria:

© Copyright IBM Corp. 2008, 2009

– Output class
– Job ID
– Forms specification
– Destination (LOCAL, or remote workstation name)
– The name of your output writing routine

The usual technique however, for setting data set selection criteria, is to build a list of eligible SYSOUT classes for the devices that will use the external writer.

If no class is specified, then we will not start selecting any output and will wait for a MODIFY command to be issued. For example, if you have the following in process:

```
//IEFPROC  EXEC  PGM=IASXWR00,PARM='P',REGION=20K
```

No class is specified. An S XWTR will cause the external writer to be started, but idle. Then, a F XWTR,F=ABC may be specified to have the external writer select by form type. If another F XWTR,C= is issued, then everything on the output queue will be selected because a null class is specified.

How to Set Up and Start the External Writer

For an external writer to work in a JES environment, it must be defined to the system in a cataloged procedure residing in SYS1.PROCLIB; and it must be started by a START command, either from the system console or from within a program.

There are two ways to set up the criteria by which the external writer selects output. The PARM= field (described below) allows the specification of SYSOUT classes to select. Other criteria (such as forms, destination, writer name, jobid and also SYSOUT class) must be specified using the MVS MODIFY command. See *z/OS MVS System Commands* for a complete description of the modify command as it applies to the external writer.

The External Writer Cataloged Procedure

The IBM-supplied external writer is defined and invoked by the cataloged procedure named XWTR, which can serve as the base or a model for a procedure you would write for your own output writer.

XWTR contains one step and consists of two JCL statements:

- The EXEC statement specifies the name of the external writer program to be executed.
- The DD statement, which **must** be named IEFRDER as shown below, defines the output data set.

Following is the actual XWTR procedure:

```
//IEFPROC  EXEC   PGM=IASXWR00,REGION=20K,                        X
//                PARM='PA,,'
//IEFRDER  DD     UNIT=TAPE,VOLUME=(,,,35),                       X
//                DSNAME=SYSOUT,DISP=(NEW,KEEP),                  X
//                DCB=(BLKSIZE=133,LRECL=133,BUFL=133,            X
//                BUFNO=2,RECFM=FM)
```

The EXEC Statement

The generalized format for the EXEC statement is:

```
//IEFPROC  EXEC  PGM=IASXWR00[,REGION=nnnnnK,]
//              [PARM='cxxxxxxxx[,seprname]]'
//              [PARM='cxxxxxxxx[,seprname][,CERTIFY=Y|N]]'
```

The stepname must be IEFPROC, as shown. The parameter requirements are as follows:

PGM=IASXWR00
The name of the external writer load module. It must be IASXWR00, as shown.

REGION=nnnnnK
This parameter specifies the region size for the external writer program. The value nnnnn is a 1- to 5-digit number that is multiplied by 1K (1024 bytes) to designate the region size. The region size can vary according to the size of buffers and the size of your output writing routine. Insufficient region size will cause the external writer to abend.

[PARM='cxxxxxxxx[,seprname][,CERTIFY=Y|N]]'
[PARM='cxxxxxxxx[,seprname][]'
Specifies output writer routine characteristics.

c An alphabetic character, either **P** (for printer) or **C** (for card punch), that specifies the control characters for the class of output the output writing routine will process.

xxxxxxxx
From one to eight (no padding required) single-character SYSOUT class names. These characters specify the classes the output writing routine will process and establish the priority for those classes, with the highest priority at the high-order (leftmost) end of the character string.

Note: If the START command includes class name parameters, they override all of the class names coded here. If you do not code class names on the procedure EXEC statement or the START command, then the external writer will wait for a MODIFY command from the operator before processing any output.

seprname
This is the name of the output separator routine to run with the output writing routine. If you omit this subparameter, no output separator pages are produced.

IEFSD094 is the IBM-supplied output separator routine. If you write your own output separator routine, and name it here, put it in SYS1.LINKLIB (or a library concatenated to LINKLIB through a LNKLSTxx member of parmlib).

Note: Do not use *CERTIFY* as the name of your output separator routine. For your separator routine to be invoked, you must either code its name on this subparameter or name it in the parameters on the START command.

CERTIFY=Y|N
Indicates whether (Y) or not (N) you want to ensure that all data is safely written to the output device. CERTIFY=Y indicates that data will **not** be lost even if the external writer abnormally terminates, such as would occur if the output data set becomes full.

Notes:

1. The use of the certify function increases processing overhead and decreases efficient space utilization by the output data set.
2. If you write your own output writing routine and require the CERTIFY function, you need to modify your code to be similar to that supplied by

IBM. Refer to “Using the CERTIFY Function” within section “Programming Considerations for the Output Writing Routine” on page 35 for further details.

3. IBM cannot guarantee data set integrity when writing multiple data sets to a PDS/PDSE member even if you specify CERTIFY=Y. See “Functions of the Output Writer Routine” on page 32 for further details.

The DD Statement

In this statement, you must define the output data set that the external writer will use. The generalized format for the DD statement is:

```
//IEFRDER  DD  UNIT=device,LABEL=(,type),                          X
//             VOLUME=(,,,volcount),DSNAME=anyname,                X
//             DISP=(NEW,KEEP),DCB=(list of attributes),           X
//             UCS=(code[,FOLD][,VERIFY]),                         X
//             FCB=(image-id[,ALIGN]|[,VERIFY])
```

The ddname must be IEFRDER, as shown. The parameter requirements are as follows:

UNIT=device
This specifies the printer, tape, card punch, or DASD device on which the output data set is to be written.

LABEL=type
This describes a data set label, if one is needed (for tape data sets only). If this parameter is omitted, a standard tape label is used.

VOLUME=(,,,volcount)
Needed for tape data sets only, this parameter limits the number of tape volumes that this external writer can use during its entire operation.

DSNAME=anyname
This specifies a name for the output data set, so later steps in the procedure can refer to it. The data set name is required for the disposition of KEEP.

DISP=(NEW,KEEP)
The disposition of KEEP prevents deletion of the data set (tape and DASD only) at the end of the job step.

DCB=(list of attributes)
The DCB parameter specifies the characteristics of the output data set and the buffers. The BLKSIZE and LRECL subparameters are always required. The BUFL value, if you do not code it, is calculated from the BLKSIZE value. Other subparameter fields may be coded as needed; if they are not, the defaults are the QSAM default attributes. These are:

BUFNO—
Three buffers for the 2540 punch; two buffers for all other devices.

RECFM—
U-format, with no control characters.

TRTCH—
Odd parity, no data conversion, and no translation.

DEN—
Lowest density.

OPTCD—
Printer data checks are suppressed, and “select translate table” characters are printed as data. The IBM external writer does not support OPTCD=J, a printer dependent specification.

UCS=(code[,FOLD][,VERIFY])
This specifies the code for a universal character set (UCS) image to be loaded into the UCS buffer.

FOLD causes bits 0 and 1 to be ignored when comparing characters between the UCS and print line buffers, thereby allowing lowercase alphabetic characters to be printed (in uppercase) by an uppercase print chain or train.

VERIFY causes the specified UCS image to be printed for verification by the operator.

The UCS parameter is optional, and is valid only when the output device is a 1403, a 3211, or a 3203-5 printer.

FCB=(image-id[,ALIGN]|[,VERIFY])
This causes the specified forms control buffer (FCB) image to be loaded into the FCB. ALIGN and VERIFY are optional subparameters that allow the operator to align forms. In addition, VERIFY causes the specified FCB image to be printed for visual verification. The FCB parameter is valid only for a 3203-5, 3211, or 3800 printer; otherwise, it is ignored.

See *z/OS MVS JCL Reference* for more information on the parameters mentioned here.

Special Printer Output Considerations: To process output jobs that require special chains for printing, you should have specific classes for each different print chain. You can specify the desired chain in your output writer procedure, and when that output writer is started, the chain will be loaded automatically. (Printers used with special chains should be named with esoteric group names as defined at system installation.) See *z/OS HCD Planning* for information on the eligible device table.

Following is an example of the JCL needed to define a special print chain in a cataloged procedure for an external writer.

```
//IEFPROC EXEC PGM=IASXWR00,REGION=20K,PARM='PDEG,IEFSD094'
//IEFRDER DD   UNIT=SYSPR,DSNAME=SYSOUT,FCB=(STD2,ALIGN),           X
//             UCS=P11,DISP=(,KEEP),                                X
//             DCB=(BLKSIZE=133,LRECL=133,BUFL=133,                 X
//             BUFNO=2,RECFM=FM)
```

In this example, the UCS DD parameter requests the print chain alias for data sets in the SYSOUT classes D, E, and G.

If the output device is a 3211 or a 3203-5, a UCS or FCB image can be loaded dynamically between the printing of data sets. Therefore, you can specify a mixture of data sets using different images in a single output class for this device. This will probably require mounting trains and changing forms, however, so it might not be desirable.

When the output device is a 1403 or 3800, the UCS image or 3800 attributes are specified at START XWTR time; they cannot be changed until the writer is stopped. Therefore, all data sets within an output class must be printed using the same train.

The FCB image is ignored when the 1403 printer is the output device.

External writer output to an IBM 3800 Printing Subsystem can also make use of the CHARS, COPIES, FLASH, and MODIFY JCL parameters on the DD statement. For

information about using these parameters, see *3800 Printing Subsystem Programmer's Guide*. The coding rules and defaults are documented in *z/OS MVS JCL Reference*.

How the IBM-Supplied Output Writer Routine Works

As stated previously, there are two main pieces to the external writer that can be modified or replaced by the installation: The output writer routine and the output separator routine. When the external writer is called, it gets control in module IASXWR00. The main function of IASXWR00 is to initialize a 7-word parameter list for the use of the other external writer routines.

The parameter list contains information about the output device and DCB addresses for each data set. The input data set to the external writer was a job's output data set which met the criteria for processing by the external writer. This input data set is not yet open.

The format of the external writer parameter list is as follows:

Table 2. External Writer Parameter List

Byte	Meaning
0	Used to describe the type of output device **Bit** **Meaning** **011. ....** 2540 punch unit **001. ....** 1403, 3203-5, 3211, or 3800 printer device **010. ....** tape device with punch-destined output **000. ....** tape device with printer-destined output
1-3	Reserved for IBM use
4-7	Address of the DCB for the opened output data set where the external writer will put the input records
8-11	Address of the DCB for the input data set, from which the external writer will obtain logical records.
12-15	Address of the cancel ECB for writer routine subtask (IEFSD087 or user-written writer routine).

The switches indicated by the three high-order bit settings in byte 0 can be used in translating control character information from the input records to the form required by the output device.

Functions of the Output Writer Routine

When you specify IEFSD087 as the output writer on the SYSOUT= parameter of the DD statement, or when you leave this parameter blank, control will pass to the IBM-supplied output writing routine in module IEFSD087, and it:

- Issues an OPEN (J type) for the input data set previously taken from the JES spool.
- Provides its own SYNAD error-handling routine on behalf of both the input and output data sets. See *z/OS DFSMSdfp Advanced Services* for more information about OPEN-J and SYNAD.

- Reads the input data set, using the locate mode of the GET macro.
- Calls a subroutine to handle ANSI and machine control character differences and to handle conversions between the input records and the output data set.
- Calls a routine to write records to the output data set, using the locate mode of the PUT macro.
- Closes the input data set after it has been read.
- Provides accounting support by updating fields in the SMF type 6 record for the input data set.
- Frees the buffer associated with the input data set (by issuing the FREEPOOL macro).
- Returns control to the main logic control module of the external writer, IASXWR00, using the RETURN macro and setting a return code.

Note: When writing multiple data sets to a PDS/PDSE member, if the 'end-of-output-data-set' condition arises, the output external writer cannot guarantee (even if you set CERTIFY=Y as a startup parameter) that all records will be saved. The current data set is saved, but all records written up to that point cannot be recovered in the case of writing to a PDS/PDSE member.

The following diagram shows the general flow of the IBM-supplied output writing routine:

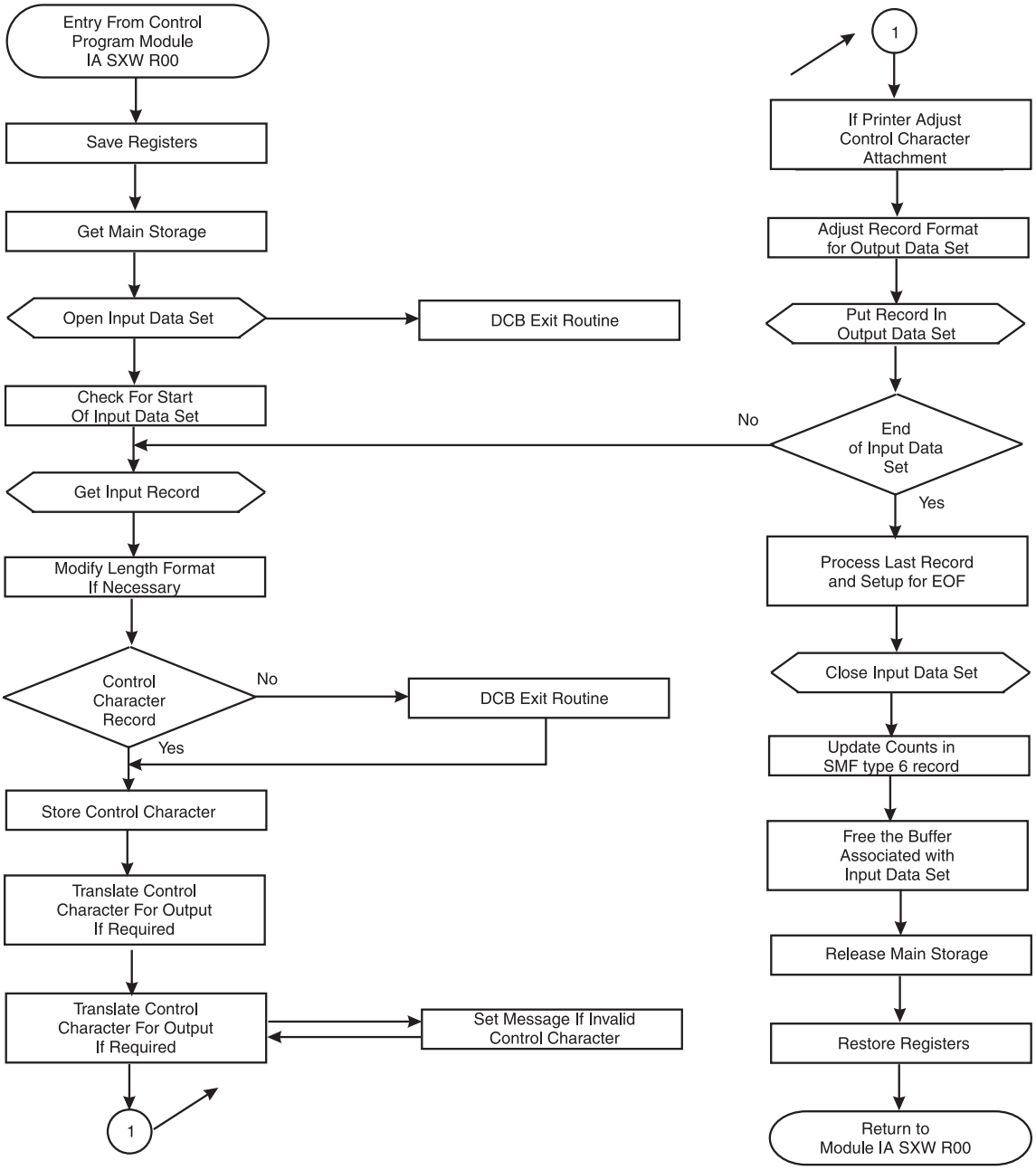


Figure 3. General Flow of IBM's Output Writing Routine

How to Add Your Own Output Writing Routine

If you want to add your own output writing routine to the external writer, consider whether your routine will be needed more often than the IBM-supplied routine. If your routine will be invoked to write **most** of the external writer's output, you might want to replace the IBM-supplied with your own routine, so that your routine will be called by default. You can retain the IBM-supplied routine by renaming it to an alias.

Replacing the IBM-Supplied Routine

You can replace the IBM-supplied routine with your routine by:

- Renaming the IBM-supplied routine (IEFSD087) to an alias.
- Naming your routine IEFSD087
- Installing your routine in SYS1.LINKLIB

The external writer will call your routine by default. You can request the IBM-supplied routine by coding its alias on the SYSOUT= parameter. For example:

```
//MYDATA  DD  SYSOUT=(H,IBMWRITR)
```

where IBMWRITR is the alias you've given to the IBM-supplied routine.

Do not code STDWTR as a writer name. STDWTR and INTRDR are and are reserved for JES used as a parameters in the MVS operator's MODIFY command.

In a JES3 system, do not code NJERDR as a writer name. NJERDR is reserved for JES3.

Using Your Routine for Only Certain Jobs

If you plan to have your routine invoked for only certain jobs, give it a unique name, and install it in SYS1.LINKLIB or an authorized library concatenated to LINKLIB through a LNKLSTxx member of SYS1.PARMLIB. To have your routine invoked, the end-user must specify its name on the SYSOUT= parameter of the job's DD statement. When your routine is not specified, the system calls the IBM-supplied routine by default.

Coding Conventions for the Output Writing Routine

In order for your output writing routine to work in the external writer, it must observe the following conventions:

- If you put your routine in LPA so that you only use one copy for all external writers, then it must be reentrant. If you use a STEPLIB and each writer can have its own copy, then the code need not be reentrant.
- The routine must use standard entry and exit linkage, saving and restoring its caller's registers.

  At entry, register 1 points to the parameter list and register 13 points to an 18-word save area.
- The routine must issue an OPEN-J macro to open the desired input data set. The data set to which the output writing routine will write records is opened before the routine is loaded.
- The routine must use the GETMAIN and FREEMAIN macros, or the STORAGE macro, to acquire and release any necessary storage.
- The routine must return control to its caller through the RETURN macro, in the same addressing mode it was called in. It must put a return code in register 15:
  - A return code of 0 indicates that the data set was processed successfully to the output device.
  - A return code of 8 indicates that the routine was unable to process the data set because of an output error.

Programming Considerations for the Output Writing Routine

In addition to the coding conventions, consider the following when writing your own output writing routine:

- **Obtaining Storage for Work Areas:** Using the GETMAIN or STORAGE macro, the output writing routine should obtain storage in which to set up switches and save record lengths and control characters.
- **Processing an Input Data Set:** To process a data set, the writing routine must get each record individually from the input data set, transform (if necessary) the record format and the control characters to conform to the output data set's requirements, and put the record in the output data set. Consider each of these tasks individually:
  1. Before the routine actually obtains a record from an input data set, the routine must provide a way to handle the two forms of record control character that are allowed in an output data set, if the output device is a printer.

     Most printers are designed so that, if the output data set records contain machine control characters, a record (line) is printed before the effect of its control character is considered. However, if ANSI control characters are used in the output records, the control character's effect is considered before the printer prints the line. Thus, if the input data sets do not all have the same type of control characters, you will need to avoid overprinting the last line of one data set with the first line of the next.

     When the input records have machine control characters and the output records are to have ANSI control characters, the standard (IBM-supplied) output writing routine produces a control character that indicates one line should be skipped before printing the first line of output data.

     When the input records have ANSI control characters and the output records are to have machine control characters, the standard writing routine prints a line of blanks before printing the first actual output data set record. Following this line of blanks, the printer generates a one-line space before printing the first record.

     Depending upon the characteristics of the printers in your installation, you will probably want your output writer to perform some kind of "printer initialization" like that outlined here.
  2. After the output writing routine has properly opened the input data set and has completed any necessary printer initialization, it must obtain records from the input data set.

     The standard output writing routine uses the locate mode of the GET macro. If you use this macro, you will need to check the MACRF field of the input data set's DCB to see if GET in locate mode is allowed. If not, you can override the MACRF parameter on the GET macro itself. See *z/OS DFSMSdfp Advanced Services* for information on coding and using all the QSAM macros.

     In a JES2 system, the padding is done automatically.

     In a JES3 system, if blank truncation is in effect (TRUNC=YES specified on the BUFFER initialization statement or the SYSOUT initialization statement), then fixed format records will not be padded with blanks. You need to do the padding manually.
  3. Having obtained a record from the input data set, the output writing routine must now make sure that its format and control character are compatible with the requirements of the output data set.

     Because the output data set is already opened when the output writing routine is entered, your routine will have to adhere to the established conventions.

The standard output writing routine uses the PUT macro in the locate mode to write records to the output data set. For fixed-length output, it obtains the record length for the output data set from the DCBLRECL field of the DCB.

If an input record is longer than the length specified for the output records, the standard output writing routine truncates the input record from the right.

If an input record is shorter than the length specified for the output records, the standard output writer left-justifies the input record and pads the field with blanks.

When the output record length is variable and the input record length is fixed, the standard output writer adds control character information (if necessary) and variable record control information to the output record. Control character information is one byte, and record control information is four bytes long. Both additions are at the high-order end of the record.

If the output record is not at least 18 bytes long, the standard output writer pads it on the right with blanks. The standard output writer also adjusts the length of the output record to match the length of the output buffer.

4. When the output writing routine has successfully adjusted the input and output records, it can read the input data set until end of data. At that point, you need to consider another aspect of input data set processing: what is going to happen to the last input record?

   The standard output writer handles output to either card punch or printer, as required; your routine could also send output to an intermediate tape or DASD device. Depending upon the kind of device, the last few records obtained from the input data set will receive different treatment.

   It might happen that all the records from a given data set are not available on the output device until the output of records from the next data set is started, or until the output data set is closed. However, if you specify CERTIFY=Y as a start-up parameter, the records can be made available on the output device after all the input records have been written by the output writing routine. When the output data set is closed, the standard output writer automatically puts out the last record of its last input data set.

   For Punch Output: When a card punch is the output device, the last three output cards could still be in the machine when the input data set is closed.

   To put out these three records with the rest of the data set, and with no breaks, the standard output writer provides for three blank records following the actual data set records.

   For Printer Output: When a printer is the output device, the last record of the input data set is not normally put in the output data set at the time the input data set is closed.

   To force out this last record, the standard output writer generates a blank record to follow the last record of the actual data set.

- **Using the CERTIFY Function**: The CERTIFY function can be specified as a parameter when starting the external writer. It guarantees that data will not be lost when an abend condition occurs. If the CERTIFY function is not enabled, it is possible that records written to the QSAM buffers, but not yet written to the output device, can be purged before they are recorded during an abend situation. This is possible, for example, when the output data set is not large enough to contain the data, resulting in an 'end-of-volume' (*x*37) abend.

  The CERTIFY function flushes the buffers each time an input data set is processed, and makes that data available on the output device. However, while guaranteeing data integrity, this function might also increase processing overhead and decrease efficient output data set space utilization. Also, note that even with

CERTIFY=Y, data integrity might be compromised in the case of writing multiple JES data sets to a partitioned dataset (PDS) or PDSE in an abend situation.

If you need to use the CERTIFY function, you need to modify your output writing routine to be similar to that supplied by IBM. Refer to module IEFSD087 for an example of how to enable the CERTIFY function. Basically, you must do the following:

1. In the external writer main task, force the number of buffers for the output data set to 1. That is, set DCBBUFNO=1.
2. In the output writing routine after all records are PUT to the data set, perform a second PUT to write a blank line (a 'transition record') after the data to separate the data sets.
3. In the output writing routine, commit the data to the output device.

- **Closing Input Data Set(s)**: After the standard output writer finishes putting out the records of an input data set, it closes the data set before returning control to the calling module. All input data sets must be closed.

  After closing the input data set, it is recommended that your output writer free the buffer pool associated with the input data set (by issuing the FREEPOOL macro).
- **Releasing Main Storage**: The output writing routine should release the storage it acquired, using the FREEMAIN or STORAGE macro, before returning to its caller.
- **Handling Errors**: The routine must put a return code into register 15 before returning to its caller using the RETURN macro.

  The standard output writer sets a return code of 8 if it terminates because of an unrecoverable error on the output data set; otherwise, the return code is 0. The output writing routine must handle input errors itself.

How the IBM-Supplied Output Separator Routine Works

The second major part of the IBM-supplied external writer routine is the output separator routine. Any output processing to a punch or printer must include a means of separating one job from another within the continuous deck or listing. When you specify IEFSD094 on the PARM= parameter on the EXEC card of the external writer, the IBM-supplied output separator routine writes separation records to the output data set prior to the writing of each job's output.

You can modify this separator routine to suit your installation's needs, or you can create your own routine. IBM's version does the following:

- **For Punch-Destined Output:** The separator routine provides three specially-punched cards (deposited in stacker 1) prior to the punch card output of each job. Each of these cards is punched in the following format:
  Columns 1 to 35
  blanks
  Columns 36 to 43
  job name
  Columns 44 to 45
  blanks
  Column 46
  output classname
  Columns 47 to 80
  blanks
- **For Printer-Destined Output**: The IBM-supplied separator routine provides three specially-printed pages prior to printing the output of each job. Each of these separator pages is printed in the following format:

- Beginning at the channel 1 location (normally near the top of the page), the job name is printed in block characters over 12 consecutive lines. The first block character of the 8-character job name begins in column 11. Each block character is separated by 2 blank columns.
- The next two lines are blank.
- The output classname is printed in block characters covering the next 12 lines. This is a 1-character name, and the block character begins in column 35.
- The remaining lines, to the bottom of the page, are blank.
- In addition to the block characters, a full line of asterisks (*) is printed twice (that is, overprinted) across the folds of the paper. These lines are printed on the fold *preceding* each of the three separator pages, and on the fold *following* the third page. This is to make it easy for the operator to separate the job output in a stack of printed pages.

To control the location of the lines of asterisks on the page, the IBM-supplied separator routine requires that a channel 9 punch be included (along with the channel 1 punch) on the carriage control tape or in the forms control buffer (FCB). The channel 9 punch should correspond to the bottom of the page. The printer registration should be offset to print the line of asterisks on the fold of the page.

The IBM-supplied separator routine makes no provision for the 3800 printing subsystem; if you use it on a 3800, the FCB must locate a channel 9 punch at least one-half inch from the paper perforation.

Separator Routine Parameter List: IBM's external writer provides its separator routine with a 7-word parameter list of necessary information. When the separator routine receives control, register 1 contains the address of the parameter list, which contains the following:

Table 3. Separator Routine Parameter List

Byte	Meaning
0	Used to describe the type of output device **Bit** **Meaning** **011. ....** 2540 punch unit **001. ....** 1403, 3203-5, 3211, or 3800 printer device **010. ....** tape device with punch-destined output **000. ....** tape device with printer-destined output
1-3	Not used, but must be present
4-7	Address of the DCB for the opened output data set where the external writer will put the input records
8-11	Address of an 8-character field containing the job name
12-15	Address of a 1-character field containing the output class name
16-19	Address of the data set PROC name in the SSOB extension.
20-23	Address of the data set STEP name in the SSOB extension.
24-27	Address of the data set DD name in the SSOB extension.

The parameter list points to a DCB; this DCB is established for the QSAM output data set, which is already open when the separator program receives control.

The address of the job name and the address of the output classname are provided in the parameter list so they can be used in the separation records the separator routine writes.

Output from the Separator Routine: A separator routine can write any kind of separation identification. IBM supplies a routine that constructs block characters. (See "Using the Block Character Routine" on page 41.) The separator routine can punch as many separator cards, or print as many separator pages, as necessary.

The output from the separator program must conform to the attributes of the output data set. To find out what these attributes are, examine the open output DCB pointed to by the parameter list. The attributes are:

- Record format (fixed, variable, or undefined length)
- Record length
- Type of carriage control characters (machine, ANSI, or none)

For printer-destined output, a separator routine can begin its separator records on the same page as the previous job output, or skip to any subsequent page. However, the separator routine should skip at least one line before writing any records because in some cases the printer is still positioned on the line last printed.

After completing the output of the separation records, the separator routine should write sufficient blank records to force out the last separation record. This also allows the error exit routine to obtain control if an uncorrectable output error occurs while writing the last record. One blank record, for printer-destined output, and three blank records, for punch-destined output, are sufficient to force out the last record.

How to Add Your Own Output Separator Routine: If you write your own separator routine, it must conform to the following requirements:

- The routine must have a unique name and this name must be specified in the PARM= parameter on the EXEC card for the external writer. If you do not specify the name of an output separator routine, no separator records are written.

  Note: Do not use *CERTIFY* as the name of your output separator routine. Refer to "The EXEC Statement" on page 28 for details on specifying an output separator routine on the PARM= parameter.
- If you want to replace the standard IBM routine, then name your routine IEFSD094. This should reside in SYS1.LINKLIB or in a library concatenated to LINKLIB through a LNKLSTxx member of SYS1.PARMLIB.
- The routine must use standard entry and exit linkages, saving and restoring its caller's registers, and returning to its caller through the RETURN macro, with a return code in register 15.
- The routine must use the QSAM PUT macro in locate mode to write separation records to the output data set.
- The routine must use the GETMAIN and FREEMAIN macros, or the STORAGE macro, to obtain and release the storage required for work areas.
- The routine must establish its own synchronous error exit routine, and place the exit address in the DCBSYNAD field of the output DCB. The error routine will receive control during output writing in case of an uncorrectable I/O error; it must set a return code of 8 (binary) in register 15 to indicate an unrecoverable output error.

  If the separator routine completes processing successfully, it must set a return code of 0 in register 15, before returning to its caller.

Note: The separator routine receives control in problem-program state, but with a protection key of 0. Therefore, the routine must ensure data protection during its execution.

Using the Block Character Routine: For printer-destined output, the separator routine can use an IBM-supplied routine to construct separation records in a block character format. This routine is a reentrant module named IEFSD095 that resides in the module library SYS1.AOSB0.

The block character routine constructs block letters (A to Z), block numbers (0 to 9), and a blank. The separator routine furnishes the desired character string and the construction area. The block characters are constructed one line position at a time. Each complete character is contained in 12 lines and 12 columns; therefore, a block character area consists of 144 print positions. For each position, the routine provides either a space or the character itself.

The routine spaces 2 columns between each block character in the string. However, the routine does not enter blanks between or within the block characters. The separator routine must prepare the construction area with blanks or other desired background before entering the block character routine.

To invoke the IBM-supplied block character routine, the IBM-supplied separator routine executes the CALL macro with the entry point name of IEFSD095. Since the block characters are constructed one line position at a time, complete construction of a block character string requires 12 entries to the routine. Each time, the address of a 7-word parameter list is provided in register 1.

The parameter list contains the following:

Table 4. Block Character Routine Parameter List

Byte	Meaning
0-3	This fullword is the address of a field containing the desired character string in EBCDIC format.
4-7	This fullword is the address of a field containing the line count as a binary integer from 1 to 12. This represents the line position to be constructed on this call.
8-11	This word is the address of a construction area where the routine will build a line of the block character string. The required length in bytes of this construction area is 14n-2, where n represents the number of characters in the string.
12-15	This word is the address of a fullword field containing, in binary, the number of characters in the string.
16-19	Address of the data set PROC name in the SSOB extension.
20-23	Address of the data set STEP name in the SSOB extension.
24-27	Address of the data set DD name in the SSOB extension.

Appendix. 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/OS 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/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Vol I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS information

z/OS information is accessible using screen readers with the BookServer/Library Server versions of z/OS books in the Internet library at:

`http://www.ibm.com/systems/z/os/zos/bkserv/`

© Copyright IBM Corp. 2008, 2009

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. 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 IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described 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:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who want to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs

© Copyright IBM Corp. 2008, 2009

and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

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

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the intended programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's intended programming interface.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming Interface Information

JES Application Programming primarily documents information that is NOT intended to be used as Programming Interfaces of z/OS.

JES Application Programming also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of z/OS. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

- Programming Interface information
- End of Programming Interface information

Policy for unsupported hardware

Various z/OS elements, such as DFSMS™, HCD, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

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.

Other company, product, or service names may be trademarks or service marks of others.

Index

A

accessibility 43

B

block character routine
  See output separator routine for an external writer

C

Client/Server print interface 15
  botain status 16
  compare CTOKENs 15
  create a CTOKEN 15
  data set 16
    access 16
    security 17
  event
    listen for event 18
  header page
    identify a requestor 17
configuration
  configuration 21
  defining 21
  internal reader 21

D

disability 43

E

external writer 27
  block characters to separate jobs 41
  characteristics 27
  description 27
  features of IBM-supplied version 27
  overview 27
  parameter list 32
    format and content 32
  selection criteria
    data set 28
  starting the external writer 28
  using the 3800 Printing Subsystem 31
  XWTR cataloged procedure 28
    for the 3800 Printing Subsystem 31
    modifying for special print chains 31

I

IASXWR00 module 32
IEFSD094 output separator routine 38
IEFSD095 output separator routine 41
internal reader
  example
    allocation, dynamic 22
internal reader *(continued)*
  example *(continued)*
    dynamic allocation 22
    submitting from a task 21
    submitting from job 21
  maximum number of simultaneous job streams 21
  using 21

K

keyboard 43

M

mainframe
  education ix

N

non-JES output writing routine
  See external writer

O

output separator routine
  writing 40
output separator routine for an external writer
  block character routine by IBM 41
    functions 41
    invoking 41
    module name IEFSD095 41
    parameter list 41
  description 38
  IBM-supplied output separator routine 38
  note on protection key 41
  output from the routine 40
  parameter list 39
  requirements for writing 40
    return codes 40
output writing routine 32
  coding conventions 35
  description 32, 34
  IBM-supplied routine 32
  programming considerations 35

P

performance considerations for JES2 Readers
  held internal reader 24
  Record length of SYSIN data sets 24
  SYSIN record formats 25
  use of unblocked records for SYSIN and SYSOUT data sets 24

© Copyright IBM Corp. 2008, 2009

S

separator routine
See also output separator routine for an external writer
parameter list 39
shortcut keys 43
spool data set browse 3
ACB/RPL interface 8
allocation 3
compatibility interface 7
end of processing 12
performance 12
secondary subsystem support 13
started task 23
control statements that affect the internal reader 24
discussion 22
example of conditional JCL submitted from a cataloged data set 23
example of console command to read job cataloged data set 23
example of console command to read job from tape 23
example of RDR procedure 23
example to submit from a tape through a batch job 23

T

time-sharing task
logon 22

X

XWTR cataloged procedure 28

Z

z/OS Basic Skills information center ix

Readers' Comments — We'd Like to Hear from You

z/OS
JES Application Programming

Publication No. SA23-2240-01

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 your comments via e-mail to: mhvrcfs@us.ibm.com

If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	E-mail address

Readers' Comments — We'd Like to Hear from You
SA23-2240-01

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 Corporation
MHVRCFS, Mail Station P181
2455 South Road
Poughkeepsie, NY
12601-5400

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line



Program Number: 5694-A01

Printed in USA

SA23-2240-01

