IBM BatchPipes OS/390 V2R1 Users Guide and Reference

SA22-7458-00

IBM BatchPipes OS/390 V2R1 Users Guide and Reference

SA22-7458-00

Note

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

First Edition, April, 2000

This edition applies to IBM BatchPipes OS/390 (5655-D45).

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

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

International Business Machines Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+914+432-9405
FAX (Other Countries):
Your International Access Code +1+914+432-9405

IBMLink (United States customers only): KGNVMC(MHVRCFS)
IBM Mail Exchange: USIB6TC9 at IBMMAIL
Internet: mhvrcfs@vnet.ibm.com
World Wide Web: http://www.s390.ibm.com/os390

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 book
- 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.

Contents

Figures

Notices

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by IBM, are the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

```
IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, New York  10595
USA
```

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

```
IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA
Attention: Information Request
```

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

Any pointers in this publication to non-IBM Web sites are provided for convenience only, and do not in any manner serve as an endorsement of these Web sites.

Programming Interface Information

This publication‘ documents intended Programming Interfaces that allow the customer to write programs to obtain the services of product name‘.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

- BatchPipes
- BatchPipeWorks
- BookManager

- GDDM
- IBM
- IBMLink
- MVS/ESA
- OS/390
- S/390
- VM/ESA

About This Book

This book describes IBM BatchPipes/MVS and offers guidance, reference, and diagnostic information to help all computer personnel responsible for starting and maintaining the BatchPipes workload.

Who Should Use This Book

The book is intended for system programmers and application developers Who use the BatchPipeWorks function:

- With batch jobs that are part of a BatchPipes pipeline
- With batch jobs that are not part of a BatchPipes pipeline
- In the TSO environment, as part of a REXX exec and as a command on the TSO/E command line.

How to Use This Book

The book is organized as follows:

- Section 1, "Introduction and General Concepts" in topic 1.0 describes BatchPipeWorks terminology, how BatchPipeWorks processes its commands, and how to read the syntax diagrams in the book.
- Section 2, "Information for System Programmers" in topic 2.0 describes, planning tasks, initializing and diagnosing BatchPipes.
- Section 3, "Information for Operations Staff" in topic 3.0 describes, in reference form, the controlling, scheduling considerations, and monitoring, BatchPipes.
- Section 4, "Information for Application Developers" in topic 4.0 describes, in reference form, the planning tasks, making changes to JCL or dynamic allocation, planning recovery, security and changing application to use BatchPipes.

Six Appendixes follow the chapters:

- Appendix A, "SMF Record Description", describes programming interface information.
- Appendix B, "BatchPipes Query Service", describes a general-use programming interface and associated guidance information.
- Appendix C, "BatchPipes Version Information", describes a programming interface that allows you to verify the version of the BatchPipes code.
- Appendix D, "BatchPipes Messages", describes the messages associated with BatchPipes.
- Appendix E, "BatchPipes Codes", describes the codes used by BatchPipes.
- Appendix F, "BatchPipes Codes", describes syntax diagrams.
- Appendix G, "Enabling the BatchPipes Dynamic Exit", describes the dynamic exit used with BatchPipes.

Where to Find More Information

Where necessary, this book references information in other books, using shortened versions of the book title. The following tables show the complete titles and the order numbers of books you might need while you are using this book:

Figure 1. Basic Material: Unlicensed Publications

Publication Title	Form Number
IBM BatchPipes OS/390 V2R1 Introduction	GA22-7459
IBM BatchPipes OS/390 V2R1 Users Guide and Reference	SA22-7458
IBM BatchPipes OS/390 V2R1 BatchPipesWorks Reference	SA22-7456
IBM BatchPipes OS/390 V2R1 BatchPipesWorks User Guide	SA22-7457

Publications Useful During Installation For OS/390

Figure 2. Publications Useful During Installation For OS/390

Publication Title	Form Number
OS/390 MVS Initialization and Tuning Guide	SC28-1751
OS/390 MVS Initialization and Tuning Reference	SC28-1752
OS/390 MVS JCL Reference	GC28-1757
OS/390 MVS System Commands	GC28-1781
OS/390 Security Server (RACF) Security Administrator's Guide	SC28-1915
OS/390 SMP/E Commands	SC28-1805
OS/390 SMP/E Messages and Codes	SC28-1738
OS/390 SMP/E Reference	SC28-1806
OS/390 SMP/E User's Guide	SC28-1740
OS/390 MVS Setting Up a Sysplex	GC28-1779

Publications Useful During Installation For MVS/SP Version 5

Figure 3 (Page 1 of 2). Publications Useful During Installation For MVS/SP Version 5

Publication Title	Form Number
MVS/ESA SP V5 Initialization and Tuning Guide	GC28-1451
MVS/ESA SP V5 Initialization and Tuning Reference	GC28-1452
MVS/ESA SP V5 System Data Set Definition	GC28-1432
MVS/ESA SP V5 JCL Reference	GC28-1479
MVS/ESA SP V5 System Commands	GC28-1442
MVS/ESA SP V5 Setting Up a Sysplex	GC28-1449
SMP/E Reference	SC28-1107
SMP/E User's Guide	SC28-1302
SMP/E Messages and Codes	SC28-1108

Figure 3 (Page 2 of 2). Publications Useful During Installation For MVS/SP Version 5	
Publication Title	**Form Number**
RACF Security Administrator's Guide	SC28-1340
DFSMS/MVS V1 Program Management	SC26-4916
DFSMS/MVS V1 Utilities	SC26-4926

Other Referenced Books

Where necessary, this book references information in libraries other than the MVS/ESA library.

Short Title Used in This Book	Title	Order Number
BatchPipes Introduction	*IBM BatchPipes/MVS Introduction*	GC28-1214
DFSMS/MVS Macro Instructions for Data Sets	*DFSMS/MVS Macro Instructions for Data Sets*	SC26-4913
DFSMS/MVS: Storage Administration Guide for DFSMShsm	*DFSMS/MVS: Storage Administration Guide for DFSMShsm*	SH21-1076
RACF Security Administrator's Guide	*RACF Security Administrator's Guide*	SC28-1340

Where necessary, this book references information in libraries other than the MVS/ESA library.

Short Title Used in This Book	Title	Order Number
OS/390 MVS JCL Reference	*OS/390 MVS JCL Reference*	GC28-1757
OS/390 TSO/E REXX User's Guide	*OS/390 TSO/E REXX User's Guide*	SC28-1947
OS/390 TSO/E REXX Reference	*OS/390 TSO/E REXX Reference*	SC28-1975
VM/ESA CMS Command Reference	*VM/ESA CMS Command Reference*	SC24-5461

World Wide Web (WWW)

For the latest news about BatchPipes for OS/390, including information on how to obtain updated BatchPipes product documentation, visit us at the S/390 World Wide Web site:

```
www.s390.ibm.com/products/
```

This site is Powered by S/390.

Section I: Introduction and General Concepts

Chapter 1: Introduction to BatchPipes

IBM BatchPipes* OS/390 (BatchPipes, for short) offers a way to connect jobs so that data from one job (or many jobs) can pass through processor storage to another job (or jobs) without going to DASD or tape. It addresses a growing problem that faces installations that have batch workloads: insufficient time to complete that work. Given a batch stream with a data flow of certain characteristics, BatchPipes can significantly shorten the elapsed time of the job stream.

Installations can usually obtain the performance benefits with no change to applications' normal I/O declares, macros, or routines. BatchPipes intercepts application requests for I/O and passes the data through virtual storage.

BatchPipes, running on MVS/ESA*:

- Allows two or more jobs that formerly ran serially to run in parallel
- Reduces the number of physical I/O operations by transferring data through processor storage rather than transferring data to and from DASD or tape.

What difference does parallel processing make to my batch workload? It is common for more than one job to be running on your systems. Large tightly-coupled processing systems running with MVS are capable of having many jobs in various stages of processing at one time. BatchPipes allows existing batch applications to participate more fully in parallel processing. Jobs that once ran one after another can now run in parallel because the data records are, as they are being written, immediately available to the next job. That is, the whole sequential file does not have to be written and then closed before the next job can access the data.

What difference does keeping data in the processor make to my batch workload? When data passes from one job to another through processor storage, the transfers take place in microseconds as compared to the milliseconds required to transfer the data to and from tape or DASD.

Benefits of BatchPipes

The important benefit of running BatchPipes is that a set of jobs can run faster and process larger data volumes in the available batch window; processors are freed to run more work — perhaps extending the period of time that interactive applications are available — perhaps supporting your company's work in another time zone. In short, you can get more work done from your processors.

Additionally, keeping data in processor storage reduces the number of physical I/O operations, causes less I/O contention, and eliminates the need for the device that held the intermediate data set. This device can then serve more efficiently the other jobs that access data sets on the same device. Installations can reduce their use of tapes and DASD. Along with tape reduction comes fewer tape mounts and fewer physical tapes to purchase, maintain, and store.

Using BatchPipes

Most installations have batch job streams that use intermediate data sets such that output from one process (a job or step) is written to DASD or tape, and is available to be read by a second process after the data set closes. The following illustration shows a sequence of job-to-job data flow:

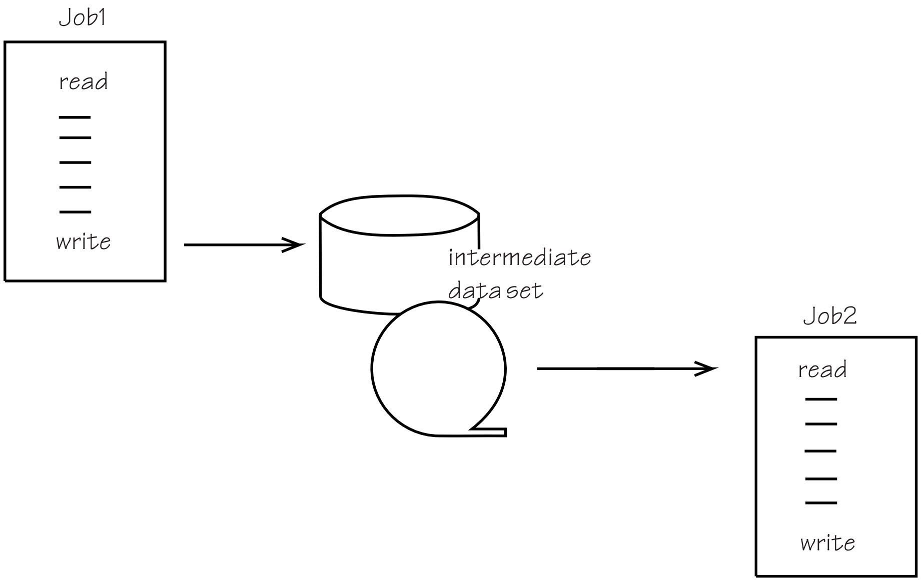


Figure 4. Two Jobs Using an Intermediate Data Set

In Figure 4, Job1 writes data to an I/O device. When all the records have been written and the data set is closed, Job2 starts. Job2 reads the data from the device. After Job2 finishes processing all the records, the data set is no longer required and can be deleted. In effect, transfer of data from one job to the other is at a data set level.

With BatchPipes, the first job does not have to close the data set before the second job begins to read from it; BatchPipes allows those two jobs to process at the same time. In Figure 5 on page 5, the two jobs run concurrently. The data moves through processor storage buffers rather than using external storage. Record-by-record, output from Job1 becomes input to Job2. Job2 does not wait for Job1 to complete before it starts. It can start reading from the processor storage buffer as soon as the first output record from Job1 is written to the buffer. You can think of the data as “flowing” from Job1 to Job2, much as water flows through a pipe.

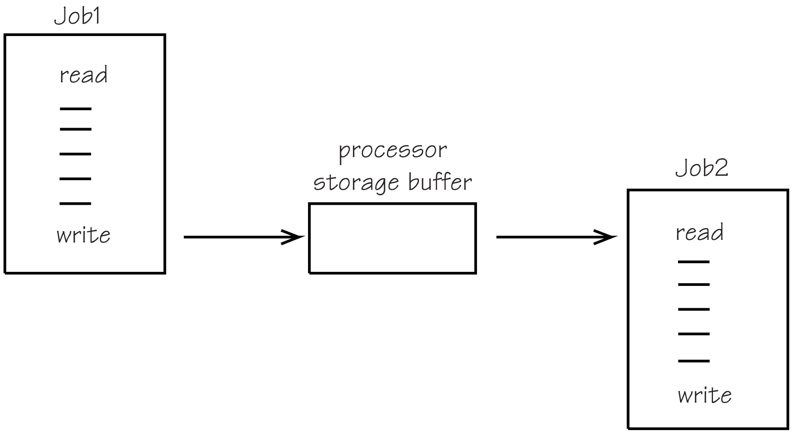


Figure 5. Two Jobs Using BatchPipes

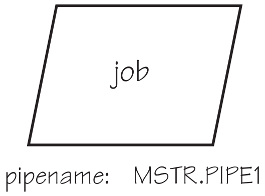
BatchPipes transfers data from one job to another job. Therefore, to get data transfer from jobstep to jobstep within the same job requires that you split the job, making the individual jobsteps into separate jobs.

Description of Data Movement with BatchPipes

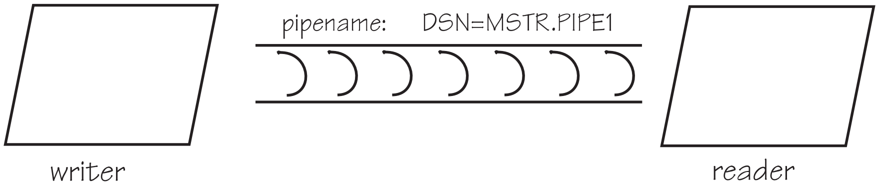
In Figure 5, you saw how the data from one job is written to processor storage, instead of to DASD or tape. In BatchPipes terms, the processor storage buffer is called a **pipe** through which data flows, always in the same direction, from a **writer** (the job that writes to the pipe) to a **reader** (the job that reads from the pipe). To understand how the writer and reader use the pipe, follow the scenario of the basic BatchPipes configuration: a writer, a pipe, and a reader. Illustrations show the pipe as capable of holding seven records; in other words, the **depth** of the pipe is seven records.

Through JCL DD statements, jobs allocate a data set and tell the system that BatchPipes is to control the movement of that data. The data will be written to a pipe, with the pipe assuming the name of the data set.

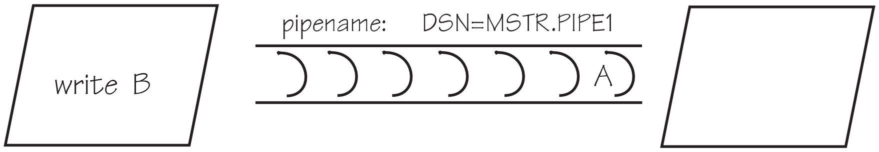
Let's proceed with the scenario. One of the two jobs starts and indicates that it is a writer to the pipe named MSTR.PIPE1. BatchPipes recognizes that a **partner** for the writer does not yet exist so it does not let the job continue. In BatchPipes terms, the writer **waits-for-open** until a reader partner appears.



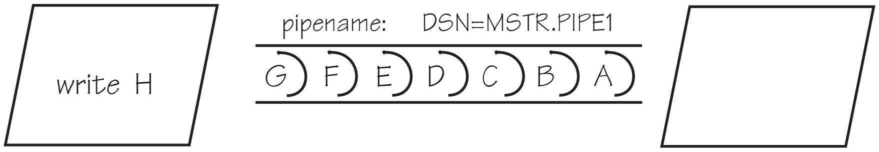
The second job starts and indicates that it is a reader from the pipe named MSTR.PIPE1. Note that the reader might have been the first to start, with the writer starting soon after. In either case, BatchPipes recognizes that the two jobs are writer/reader partners and it **builds the pipe connections**. With this action, the “pipeline” is established and the flow of data can begin.



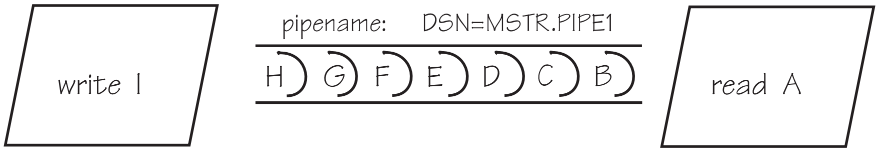
The first job begins to run and issues a write request. To show the flow of data through the pipe, the following illustrations equate a single output or input record to a letter of the alphabet, where “A” is the first record, “B” is the second record, and so forth. The illustration shows the writer adding the first record to the pipe. The reader has not issued a read request yet.



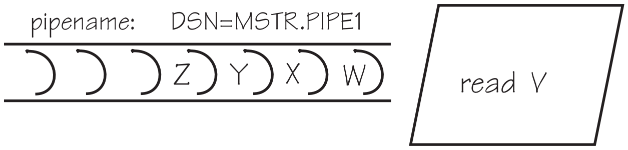
The writer fills the pipe and tries to write the record identified by “H.” Having no place to put the record, the writer **waits**.



The reader from the pipe issues its first read request and record “A” leaves the pipe. As soon as that record is removed, record “H” enters the pipe. The reader continues to read, limited by the supply of records in the pipe. The writer continues to write, limited by space available in the pipe.



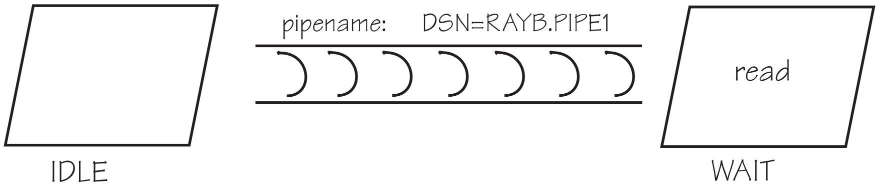
When the writer finishes writing to the pipe, it closes the data set (that is, **disconnects from the pipe**) and the reader takes the last records from the pipe and receives an end-of-file indication.



When the reader closes its data set, the pipe ceases to exist.

If the reader reads from the pipe faster than the writer writes to the pipe, the reader will wait often for the writer to add a record to the pipe. It might be that the writer does not add to the pipe over a period of time because it is performing actions that do not involve the BatchPipes subsystem. In BatchPipes terminology, the writer is **idle**. The “idling” of one of the partners causes the waiting of the other.

The following diagram shows an empty pipe. The reader is waiting for a record to be written to the pipe. The writer, however, is busy doing other work or perhaps waiting for some other action to take place.



Summary of BatchPipes Terminology

This section summarizes some of the terms you need as you use BatchPipes. The terms are important because they appear in BatchPipes monitor panels, in BatchPipes messages, and in SMF Type 91 records.

In the earlier scenario, you saw how the data from one job is written to a processor storage area known as a **pipe**, through which data flows from a **writer** to a **reader**. The jobs at either “end” of the pipe are sometimes called **writer/reader partners** or simply **partners**. The junction of a job and a pipe is a **connection**; when jobs are able to transfer data though a pipe the **connection is open**. When one of the partners has issued the OPEN, but a partner has not yet issued the OPEN, the waiting partner is in a **wait-for-open** condition.

The **name** of the pipe is the data set name. The series of job->pipe->job is called a **pipeline**, where each job is a **stage** in the pipeline. That is, job->pipe->job->pipe->job is a three-stage pipeline.

After all jobs using the pipe close, the **connections are closed** and BatchPipes deletes the pipe.

The two types of pipe connections are:

WRITE connection A job has connected to the pipe as a writer
READ connection A job has connected to the pipe as a reader.

Wait-for-allocation A writer or reader has allocated a pipe data set, but no partner has allocated the same data set.

Wait-for-open A writer or reader has opened a pipe dataset, but no partner has opened the same data set.

Wait-for-open can also mean that you have asked BatchPipes to delay building connections until a designated number of writers and a designated number of readers have opened the pipe data set; those that have already opened the pipe data set are in a wait-for-open state, awaiting the arrival of the rest.

Wait A writer has issued a write request to a full pipe or a reader issues a read request to empty pipe.

Idle A writer or reader is busy doing work that does not require the action of a BatchPipes subsystem.

Wait-for-close You have asked BatchPipes to delay deleting the pipe connection until all writers and readers using the pipe have closed the pipe data set. Then, BatchPipes deletes the pipe connections and the writers and readers can continue processing. Writers and readers that close the pipe data set and then wait for others to do the same are in a wait-for-close state.

Wait-for-EOF You have requested that BatchPipes not send an EOF indication to readers. All writers have closed the pipe data set and the last writer requested that BatchPipes not send an EOF indication. A reader is waiting at an empty pipe for an EOF indication; not receiving one, the reader waits-for-EOF.

Wait-for-termination A writer or reader has arrived at termination processing, and is waiting for its partners to do the same.

The terms BatchPipes uses to describe pipe connections and the status of the jobs using pipes are important fields as you monitor the progress of jobs. For example, STATUS command displays contain a WAIT field that identifies the elapsed time since a writer has attempted unsuccessfully to write to a full pipe. The WAIT field also identifies the elapsed time since a reader has attempted unsuccessfully to read from an empty pipe.

A Pipe with Multiple Readers or Writers

It is possible for more than one job to write to the same pipe; therefore, an end of a pipe can have more than one write connection. A pipe has as many write connections as it has writers issuing the OPEN macro for that specific “piped” data set. In a similar way, a pipe can have multiple readers from the pipe, and multiple writers ***and*** multiple readers.

To understand the use of more than one copy of a writer or reader, consider how unlikely it would be for a writer to issue write requests at exactly the same rate a reader issues read requests. **Pipe balance** refers to the ratio between the rate of writing to the pipe and the rate of reading from the pipe. The greater the difference between the rate of writing and the rate of reading, the more time the partners spend waiting and idling and not getting the work done.

To balance a pipe, your installation might add one or more writers or readers. For example, take the case where the writer writes records to the pipe twice as fast as the reader takes them from the pipe. This imbalance leaves the writer waiting much of the time, facing a full pipe. If two copies of the reader job both read records from the pipe and processed those records, the writer would be much busier and the elapsed time would improve. Figure 6 shows this pipeline, called a **one-to-many** pipeline.

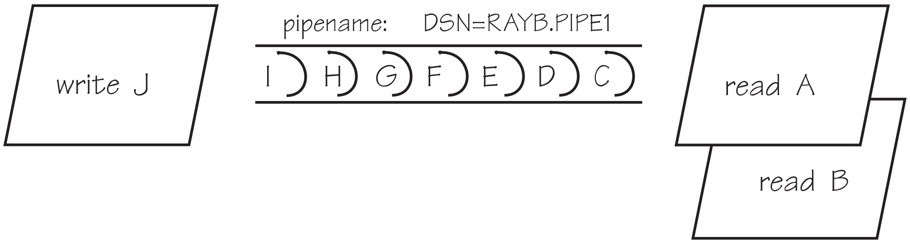


Figure 6. Example of One-to-Many Pipeline

Similarly, a pipeline can be **many-to-one** or **many-to-many**.

The process of making duplicate copies of a writer or reader is called **cloning**. For more information about how to clone a job, see “Cloning a Writer or Reader” on page 143.

Blocking with BatchPipes

So far, in describing pipe concepts, the book has described data movement as occurring one record at a time. Actually, this is the case only if the JCL that defines the data set contains a BLKSIZE value equal to the logical record length (LRECL) or where `RECFM=F` or `RECFM=V`.

As with data being written to or read from DASD, better processor performance can be achieved by accumulating records in a buffer area in the writer job's private area. BatchPipes then moves the accumulated data, known as a **block** (or physical record), to the pipe. Likewise, the reader receives the data as a block of data in a buffer in the reader job's private area.

Limits on the Use of a BatchPipes Subsystem

The limits on a single instance of a BatchPipes subsystem is:

- The number of pipe connections cannot exceed 1024. (In a pipeplex, however, this number is further limited by the size of the defining structure.)
- The number of pipes cannot exceed 512. (In a pipeplex, however, this number is further limited by the size of the defining structure.)
- The maximum pipe depth (that is, the number of buffers in the pipe) is 255. (Remember, the pipedepth parameter only allows a small portion of data to be kept in storage. The data set when piped is not permanent.) You can reduce this limit through the MAXBUFNO parameter in ASFPBPxx member of PARMLIB. However, if you use the BatchPipes PIPEDEPTH=n subsystem

parameter n can be from 1 to 32768. Pipedepth is limited by MAXBUFNO. You can only exceed this if allowed to by your security product.

There is no limit on the number of records that can flow through the pipes.

Software Requirements for BatchPipes

The required software to use BatchPipes is MVS/ESA SP Version 5.0 or higher, running with the levels of JES and DFP that support the release. The BatchPipes monitor requires that you have installed ISPF Version 3 Release 3 or higher.

Planning to Use BatchPipes

As you plan to use BatchPipes, remember that all jobs in a pipeline run concurrently. Compare the traditional batch jobstream with a BatchPipes pipeline. In the traditional batch environment, jobs within an application run sequentially, one after the other. For example, a jobstream named X consists of JobX1, JobX2, and JobX3. JobX1 runs to completion in 8 hours. All output from JobX1 is on tape or DASD before JobX2 starts. JobX2 takes 6 hours to complete; in a similar manner, JobX3 takes 8 hours. Resources, such as initiators and virtual storage, are used one job at a time. The same initiator can serve all three jobs. You handle any problems with these jobs as the problems occur — job by job — over the 22 hours (or more) that the jobs run. A timeline of the running of the three jobs might look like this:

```
 <-------JobX1-----------><-----JobX2------><--------JobX3--------->
 ___________________________________________________________________
 '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '
mid-   2     4     6     8    10    noon   2     4     6     8    10
night
```

With BatchPipes, the jobs run concurrently in shorter elapsed time. The next timeline shows the BatchPipes view of those three jobs. Compare the two timelines to see how BatchPipes changes the use of resources and the need for rescheduling jobs.

```
 <--------JobX1---------->
  <----------JobX2--------->
   <-----------JobX3--------->
 ___________________________________________________________________
 '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '
mid-   2     4     6     8    10    noon   2     4     6     8    10
night
```

With BatchPipes, each job needs its own resources and all jobs in the pipeline need all their resources **at one time**. No longer can one initiator serve the three jobs. The processing of any one of the jobs depends on the processing of other jobs on the pipeline. Likewise, termination of any of the three can cause the termination of the other two. For example, if JobX3 has to wait for an operator to mount a tape, JobX1 and JobX2 might have to wait; with JobX3 not reading from the pipe, the pipes will get full. If JobX3 terminates because of that delay, JobX1 and JobX2 terminate also. Then, when you restart JobX3, you must also start JobX1 and JobX2.

This simple example shows the need for all resources needed within the pipeline to be available at one time. It also shows the need for all jobs to be scheduled to run at the same time.

Two chapters in this book contain planning information:

- “Chapter 3: Planning Tasks for System Programmers” on page 21.
- “Chapter 11: Planning Tasks for Application Developers” on page 109.

In preparing to use BatchPipes, your installation should provide education for the computer staff that will have day-to-day responsibilities for running BatchPipes. For example, operations should understand what to expect in terms of both normal and recovery scenarios.

Chapter 2: Choosing Candidates for BatchPipes

An important activity that requires concerted efforts of application programmers, system programmers, and performance and capacity analysts is the choosing of candidate jobs to use BatchPipes. Application developers, knowing the JCL changes and application changes, can envision great improvements in elapsed time and savings of DASD and tape. Their applications, however, cannot use storage that is nonexistent or in critical supply. They need the advice of the capacity planners who understand the resources at the installation.

BatchPipes offers large improvements to some applications, but not to all. This section describes two approaches for identifying jobs to use BatchPipes; which approach you use is determined by how much work the application developers are willing to do to gain the BatchPipes improvements.

- One approach is to examine your batch jobs and identify those sets of jobs that would be **good candidates** — those sets of jobs that can use BatchPipes after a minimal JCL change. See “Criteria for Good Candidates” for a list of characteristics.
- A second approach is to identify those applications that are critical to your business and whose performance is causing bottlenecks. Estimate the work required to make the changes to use BatchPipes. To see a list of work items at an actual BatchPipes customer, see “Checklist of Planning and Implementing Tasks” on page 27.

  When you use BatchPipes to speed up a given jobstream, other jobs contending for resources might run more slowly. If other factors are equal, when it comes to a choice between candidates for BatchPipes, the jobstream that is more important to your business is generally the better candidate.

In choosing applications for BatchPipes, you need to know the criteria that determines good candidates for BatchPipes and you need to analyze the critical paths. IBM can assist you in this activity by running an SMF analysis tool that identifies the jobs that are likely to be good candidates for BatchPipes.

Criteria for Good Candidates

Examine pairs of jobs and jobsteps — potential writer/reader pairs — that meet the following criteria:

- One job or jobstep depends on completion of the other job or jobstep for its data, where a job or jobstep writes to a data set and then the other job or jobstep reads the same data set.
- The jobs process one record at a time until completion.
- The jobs use sequential QSAM or BSAM to access the data set.
- The jobs use only the basic I/O macros: GET, PUT, READ, WRITE, OPEN, and CLOSE. Note that the I/O macros that transfer data in a non-sequential order do not work with BatchPipes. Those macros include (but are not limited to) POINT, BSP, REWRITE, and NOTE.
- The intermediate data set has a record format of fixed, fixed blocked, variable, or variable blocked.

- The jobs must be capable of running at the same time.
- The jobs can run on the same image of MVS.

Considerations When Applying the Criteria

Some jobs that do not meet the criteria described above should be considered anyway. For example, one of the criteria states, "The data set is intermediate; that is, there is no need for the data set to exist after the reader uses the data." You can meet the criteria by changing the application so that it writes twice: once to a pipe and once to a physical storage device. If the application cannot be changed, insert a separate job into the pipeline to write each record twice, once to a pipe and once to an external device. This process, known as **hardening data**, is an example of a **filter job** — one that uses a pipe to change the way data flows in the pipeline. For more information about a filter that hardens data, see "Writing a Copy of Data to Tape or DASD" on page 148.

To identify whether the data is intermediate data, that is, of a temporary nature, check the job's JCL, specifically the DD statement that defines the data set to be piped. Check for instances of:

- Single or multiple jobs that create what <u>looks</u> like permanent data (OLD,KEEP) but where the data is not used for any down-stream process. For purposes of BatchPipes, these are also intermediate data sets.
- Single jobs that use true intermediate data sets. These are easily identified manually or automatically because they are created with (NEW,PASS) in one step and deleted in a later step with (OLD,DELETE).

At first examination, a data set that is defined as a generation data set (GDG) also does not meet the criteria. GDGs require catalog interactions; BatchPipes doesn't interact with the catalog in any way. If you want to use BatchPipes with such data sets, you must determine whether you need the permanent copy. If so, you can use BatchPipes by adding a hardening data filter job to the pipeline.

Jobs that are Not Good Candidates

BatchPipes does not support data sets that are read by multiple jobs in the pipeline. If two jobs use a common non-piped data set, they might use that data set in a way that would prevent the jobs from running successfully in parallel. Examples of job pairs that cannot run in parallel are:

- Certain dependencies exist between the two jobs. Examples are:
  - A second job's deletion of a data set requires that the first job has already created the data set.
  - The starting of a second job depends on an accumulation of data collected throughout the processing of the first job.
  - Either of the two jobs need exclusive use of the data set; for example, if a second job does not start until a first job tells it the number of records to process, the jobs are not good candidates.

Jobs that update a data base through a data base manager are often not good candidates for BatchPipes. Those jobs require the ability to restart at a certain point. With BatchPipes, jobs must be restarted from the beginning.

For jobs that do not meet the criteria, consider other technologies, such as VIO to expanded storage, hiperbatch*, batch local shared resources, and Sequential Data Striping. See the section in the *BatchPipes Introduction* that compares those technologies to BatchPipes.

Analyzing Critical Paths

Choosing candidates for BatchPipes requires you to analyze critical paths and identify jobs that hinder the flow of work through the system. This analysis requires knowledge of the elapsed time of jobs and job dependencies:

- Elapsed time of jobs

  Start your analysis by looking at sets of jobs that run for long periods of time. But, don't overlook jobs that run often and for short lengths of time. Speeding up several short-running jobs can improve the elapsed time of the entire job stream.

  To find job-start and job-end times, look at SMF data, SYSLOG output, and reports from job scheduling products.

- Scheduling dependencies

  If a job or jobstep depends on completion of a certain event, that job or jobstep might not meet the BatchPipes scheduling requirement that the reader and writer jobs run concurrently. Analyze external dependencies by studying the job log data that identifies gaps occurring between a writer job ending and a reader job starting. This gap can indicate, for example, that the reader job is:

  – Waiting for a data set to be closed
  – Waiting for delivery of a tape from another location
  – Waiting for another job to complete
  – Dependent on a time-of-day event.

  When writer and reader job or jobstep must both wait for external events, they still might be good candidates for BatchPipes. The decision depends on the time difference between the events and the potential time savings. For example, if the events are two hours apart and you can save four hours by running them with BatchPipes, they might be good candidates.

Sort Programs as Candidates

Is a sort program a good candidate in a pipeline? A sort program must read and process all records before it writes any records out. In other words, the SORTIN file must close and the sorting process complete before the sort program writes any records. Consider two general observations on sort programs:

- If the sort program is being used to exclude or omit records or perform similar operations not involving the actual operation of **sorting**, this may be a good candidate for a pipeline stage.

- Sort products generally can generate multiple copies of the output file. This capability makes a sort job in a pipeline an ideal spot to “harden” the data (that is, write the data to tape or disk for recovery purposes).

To answer the question about sort programs as good candidates, you need to examine your reasons for using BatchPipes with the sort program. Most likely, your objective is one of the following:

- **To reduce elapsed time by effecting concurrent pipeline processing**

  Having a sorting operation at the beginning or the end of a pipeline can be very beneficial; see Figure 7 and Figure 8 on page 17 to see examples of sorting jobs at the beginning and the end of a pipeline. A sort program in the middle of a pipeline does not reduce the elapsed time as much. Consider the pipeline that consists of three jobs: Job1, a sort program, and Job2. The SORTIN and SORTOUT processing overlaps with the jobs at both ends of the pipe:

  ```
  <--------Job1---------->
         <--------SORTIN--sorting operation--SORTOUT------->
                                          <-----------Job2--------->
  ```

  Concurrency exists during the time the sort program is reading input and writing output.

- **To avoid increasing processor usage for sorting**

  For two reasons, a sort program in a pipeline increases the use of the processor:

  – Although a sort program can choose the most effective way to access an external storage device, it cannot choose how to access a pipe.

  – Generally, the sort program is unaware of how much data it is to read from a pipe. “Considerations for Sort Programs in a Pipeline” on page 116 describes how you might tell the sort program about the size of the data.

- **To reduce tape handling**

  The use of a pipe for either sort input or sort output reduces the number of tape mounts and tapes required for the data.

- **To reduce need for temporary storage**

  Using BatchPipes with large amounts of sort data can reduce the need for allocating large SORTIN and SORTOUT data sets.

- **To avoid real I/O and reduce I/O contention**

  Because BatchPipes uses processor storage for the I/O, channel paths and real devices are not involved in the transfer of data for the input or output of a sort program. Look at the following illustrations. Figure 7 shows JOBA reading the piped output from the sort program. Although sort work data sets may still be required, no external storage devices are required for the output of the sort program.

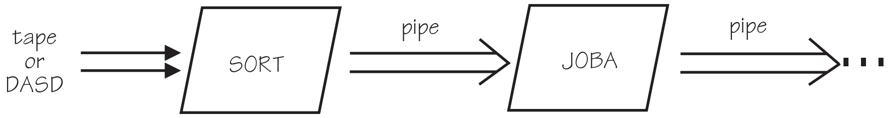


Figure 7. Reading from a Sort Program

Likewise, no external devices are required for the input of the sort program in Figure 8 that shows JOBB writing to a sort program.

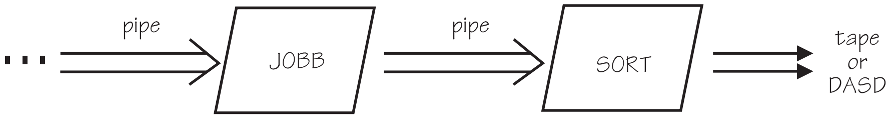


Figure 8. Writing to a Sort Program

If you are considering cloning jobs and then sorting their output, see “Cloning a Writer or Reader” on page 143.

Merge Jobs as Candidates

Is a merge job a good candidate in a pipeline? A merge job processes records very differently from a sort program. Unlike the sort program, the merge job can start writing records shortly after it reads the first record. Therefore, it can run in parallel with other jobs during the entire time it reads, processes, and writes, making it an excellent candidate for BatchPipes.

Under certain circumstances, adding a merge job to a pipeline is necessary. Often a merge job is needed immediately following a one-to-many pipeline stage. Consider the case where you have JobA processing records that are in alphabetical order. You have cloned JobB so that three copies of JobB take turns reading the records that JobA writes. If a successor job needs the records in alphabetical order, you would have to add a merge job to the pipeline to reorder the records. In this way, a merge job in a pipeline helps reduce tape mounts, external storage usage, I/O contention, and enables the jobs to gain the benefit of parallel processing.

Controlling the Use of a Pipe

BatchPipes allows you to specify some conditions under which pipe connections are built and deleted. On subparameters on the JCL SUBSYS parameter, you can:

- Specify the number of writers and readers that must open the pipe data set before BatchPipes builds pipe connections and allows data to flow through the pipe.

  Writers and readers that are first to open the pipe data set and then wait for the rest to open are in a **wait-for-open** state.

- Request that all writers and readers using the pipe close the pipe data set before BatchPipes deletes the pipe connections and allows jobs to continue processing.

  Writers and readers that are first to close the pipe data set and wait for the rest to close are in a **wait-for-close** state.

- Request that BatchPipes not send an EOF indication to readers.

  Readers that stay open with no writer partner, waiting for data and not receiving an EOF indication, are in a **wait-for-EOF** state. This reader connection can be closed by an operator issuing the BatchPipes EOF command.

- Request that BatchPipes fail a reader job that closes the pipe data set before an EOF indication arrives.
- Request that BatchPipes build a connection when only one partner opens the pipe data set and accept that user's write or read requests.

Subparameters that provide these capabilities are described in “Adding the SUBSYS Parameter to the DD Statement” on page 118.

Section II: Information for System Programmers

Section II is organized into the following chapters:

- **"Chapter 3: Planning Tasks for System Programmers"** page 21 describes the changes you can expect with BatchPipes and identifies the planning tasks you must undertake before you start to run BatchPipes.
- **"Chapter 4: Initializing BatchPipes Subsystem"** page 29 describes how to initialize BatchPipes based on your installation's workloads, resources, and requirements.
- **"Chapter 5: Diagnosing Problems Associated with BatchPipes"** page 43 helps you handle problems that might occur with jobs using BatchPipes and introduces tools to help you debug BatchPipes problems and respond to IBM* service personnel.
- **"Chapter 6: Managing the BatchPipes Subsystem"** page 49 contains information for those who initialize the BatchPipes for OS/390.

The information in this section is intended for those who might perform these tasks on behalf of BatchPipes:

- Set appropriate performance goals for BatchPipes
- Plan the use of resources
- Estimate the effect BatchPipes will have on the existing workload
- Determine how to charge the users of BatchPipes services
- Initialize a BatchPipes subsystem
- Provide recovery to protect the applications
- Automate BatchPipes operations
- Debug problems with BatchPipes.
- Defining Coupling Facility Structure

Chapter 3: Planning Tasks for System Programmers

This chapter describes planning tasks for system programmers. Running BatchPipes means changing the use of some resources at your installation. It also changes how some things work at your installation. By planning carefully, you can get the most benefit from BatchPipes and not affect how the other work in the system performs.

Summary of Planning Tasks

The following list summarizes the topics you need to address before you run BatchPipes:

- Make sure there is enough virtual storage defined for the pipes and the BatchPipes subsystem. See “Evaluating the Use of Virtual Storage” on page 22.
- Make sure you have the processor cycles that you need. See “Evaluating Processor Usage” on page 22.
- Make sure there are enough initiators defined. See “Providing Initiators for the Jobs using BatchPipes” on page 22.
- Make sure you have enough storage devices available. See “Evaluating the Use of Storage Devices” on page 23.
- Evaluate job's contention for I/O. See “Evaluating the Use of the I/O Subsystem” on page 24.
- Make sure you don't exceed the limits on the number of pipes and pipe connections. See “Evaluating the Use of Pipes and Pipe Connections” on page 24.
- Understand how BatchPipes might change the variables in your charge-back formulas. See “How BatchPipes Affects Charge-Back Formulas” on page 25.
- Understand how BatchPipes changes information returned from SMF recording. See “How SMF Recording Changes When You Run BatchPipes” on page 25.
- Assess performance considerations with BatchPipes. See “Performance Tuning for BatchPipes” on page 32 in the chapter that describes how to initialize a BatchPipes subsystem.
- Change the scheduling of jobs that use BatchPipes and, optionally, other jobs in the system. See “Chapter 8: Scheduling Considerations in a BatchPipes Environment” on page 79.

Managing Resources

BatchPipes changes the use of some resources at your installation, specifically:

- Virtual storage
- Processor
- Initiators
- Tape and DASD
- I/O subsystem
- Pipes and pipe connections

With BatchPipes, all resources for the jobs in the pipeline must be available **at one time**. Although each job might not use more resources, the use of the resources is more concentrated.

Without careful planning of resources, jobs using BatchPipes can tie up resources that other jobs need. You should encourage users at your installation to implement BatchPipes only after they have carefully planned their use of resources.

Often, jobs run faster with BatchPipes and therefore use resources for a shorter time. In some cases, however, an individual job might hold a resource for a longer period of time. For example, consider Job1 that runs for a half-hour and has a tape assigned to it during that time. With BatchPipes, it runs in a pipeline that runs for five hours. Job1 therefore holds that tape for the five hours. This fact is also true for other resources, such as DASD and initiators.

Evaluating the Use of Virtual Storage

Virtual storage is used both by the BatchPipes subsystems and by the pipes. A BatchPipes subsystem uses virtual storage in the common service area (CSA) — both above 16 megabytes and below 16 megabytes. For information about the use of virtual storage for a BatchPipes subsystem and how to reserve it, see “Reserving CSA for BatchPipes” on page 31.

The pipes themselves require some virtual storage below 16 megabytes within the application's address space. Therefore, if your application encounters system abend code `X'878'`, reduce the BUFNO value on the DD statement that defines the pipe.

No BatchPipes code resides in the link pack area (LPA).

Evaluating Processor Usage

Because the jobs in a pipeline run together, the demand for the processor increases simply to handle the work caused by the increased number of jobs running concurrently.

Providing Initiators for the Jobs using BatchPipes

Make sure you have enough initiators available to the jobs in the pipeline so that all jobs can run together. BatchPipes might not require more initiators, but it does require more **concurrent** use of initiators for those jobs.

Additional initiators are needed if you:

- Change individual jobsteps to separate jobs
- Add sort or merge programs to the pipeline
- Add a filter job to the pipeline
- Clone jobs.

In all cases, each job in the pipeline requires an initiator. If the system was close to 100% busy during the time the jobs processed without BatchPipes, you probably want to redistribute the non-BatchPipes workload so that the existing initiators cover the pipeline jobs, rather than increasing the overall number of initiators.

IBM recommends that you assign all jobs in a single pipeline to one job class and dedicate a set of initiators to that job class; this action eases the job of stopping and starting the jobs, if that is necessary. If you need to restart the BatchPipes subsystem, the least disruptive process is:

Step 1: Stop the initiators associated with the jobs using BatchPipes
Step 2: Stop the BatchPipes subsystem
Step 3: Wait for the initiators to drain
Step 4: Cancel any jobs that are still running
Step 5: Start the BatchPipes subsystem again
Step 6: Restart the initiators.

While the BatchPipes subsystem is down, don't allow any jobs to be submitted to that subsystem. That action causes a JCL error.

Evaluating the Use of Storage Devices

BatchPipes reduces the need for physical storage for temporary (intermediate) data sets. However, you still need enough devices to hold any data that is not controlled by BatchPipes, such as:

- All input data to the first job or jobs in the pipeline and output data from the last job in the pipeline
- Any devices that sort programs in the pipeline need for intermediate storage (SORTWRK data sets).

To illustrate the need for careful counting of storage devices, look at the two jobs in Figure 9. Job1 reads from four tapes and creates four tapes that become input to JobK. The tapes are demounted and then remounted between JobJ and JobK. Output from JobK goes to four tapes.

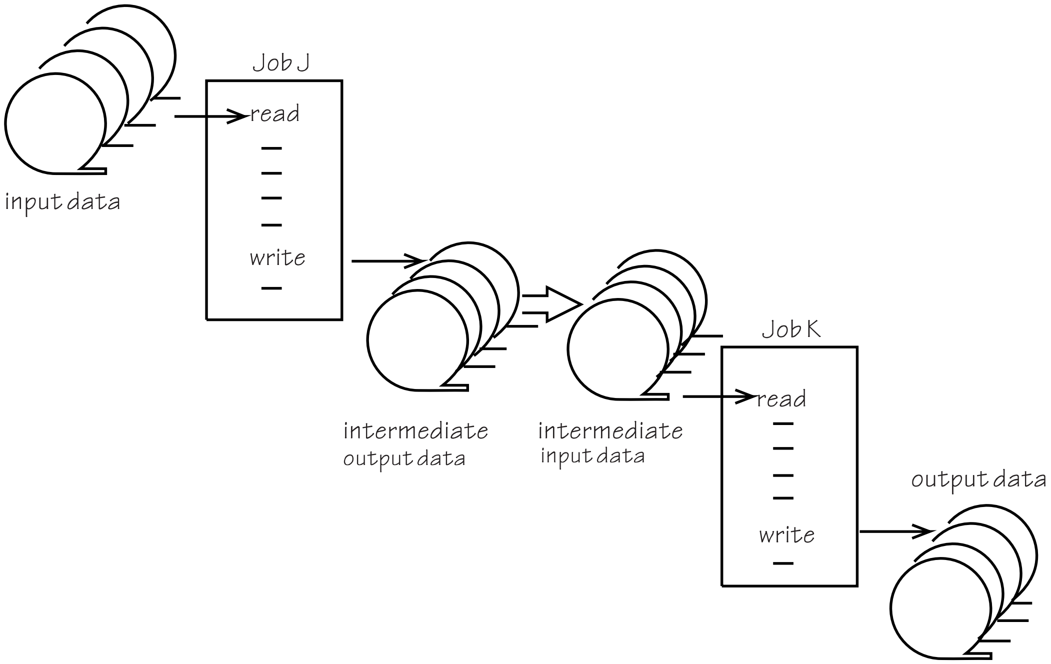


Figure 9. Customer Application without BatchPipes

When using BatchPipes the two concurrently-running jobs still use eight tapes, as Figure 10 shows.

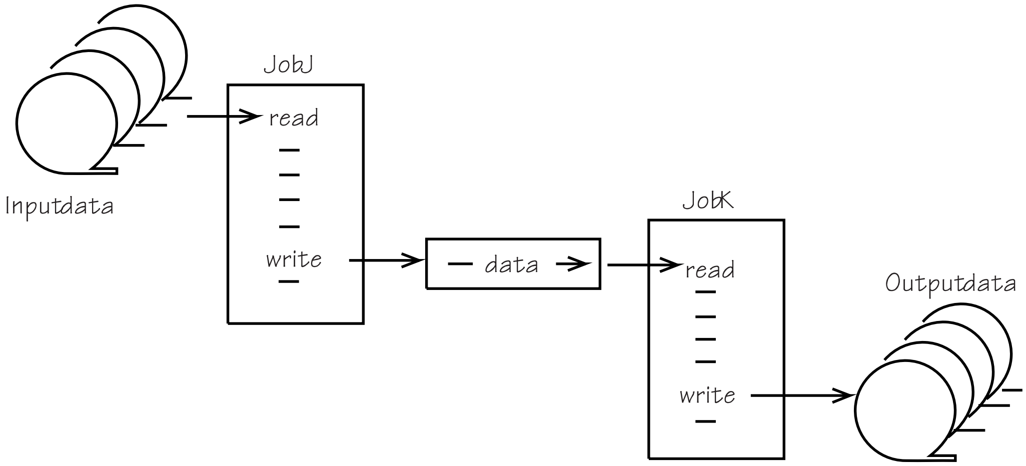


Figure 10. Customer Application with BatchPipes

The number of tape mounts decreases from 16 to eight, but the number of tape devices used stays the same.

Evaluating the Use of the I/O Subsystem

Most likely, the jobs that use BatchPipes also use data that resides on external devices. Because the jobs run in parallel, they contend with each other and with non-BatchPipes jobs for use of the I/O subsystem. Cloning jobs contributes to the likelihood of such contention. For options you can take to relieve contention, see “Reducing Contention for Data” on page 141.

Evaluating the Use of Pipes and Pipe Connections

As with other resources, pipes and pipe connections are resources that must be controlled. BatchPipes limits the number of pipes to 1024 per subsystem and the number of pipe connections to 512 per subsystem. While these limits might seem high to the user new to BatchPipes, customer experience has shown that these limits can be reached.

One way to control these resources is to make sure only those users that benefit from BatchPipes are allowed access to its subsystems.

Another way is to start up another BatchPipes subsystem. If you chose this solution:

- The jobs' DD statements must point to a specific subsystem, which means you must change those statements.
- Starting a second subsystem increases the ECSA storage required for BatchPipes code.

Additionally the following considerations apply:

- Once a job is submitted, the JCL cannot be changed
- Pipes cannot be dynamically switched from one subsystem to another.

How BatchPipes Affects Charge-Back Formulas

Whatever measurements your charge-back formulas depend on, the use of BatchPipes affects the price your end-users pay for processing. Be aware of how BatchPipes can change the variables. Figure 11 summarizes the effect that BatchPipes has on some common variables in charge-back formulas:

Figure 11. Effect of BatchPipes on Charge-Back Variables

Variable	How BatchPipes Affects the Variable
TCB time	TCB time increases. Without BatchPipes (when the I/O subsystem moved the data), significant portions of the system services (that is, the TCB time) were not charged to the address space that invoked them. With BatchPipes, the system resources that the job uses are included in the TCB time
EXCP	EXCP counts decrease. BatchPipes eliminates all EXCPs to the “piped” data sets.
Tape mounts	Tape mounts are reduced. Tape and tape mounts for the intermediate data sets that are “piped” are eliminated, along with tape mount processing and tape storage in a library.
DASD storage	DASD storage is reduced. DASD storage for the intermediate data sets that are “piped” is eliminated.
SRB time	SRB time is reduced. With BatchPipes, fewer SRBs are required to perform data movement.
Service units (SUs)	• The IOC field decreases, where IOC refers to the I/O service of an address space. • The CPU field probably increases, where CPU refers to the accumulated CPU (that is, TCB) SUs. • MSO field probably increases, where MSO refers to accumulated storage SUs.
Uncaptured CPU time	Uncaptured CPU time is reduced. Compared with accountability of non-BatchPipes jobs, accounting of CPU time is more accurate for BatchPipes jobs.
Virtual storage usage	The usage stays about the same. The use of virtual storage with BatchPipes is approximately the same as before BatchPipes.
Working set size	BatchPipes either does not affect the size or increases it. Because processing of the job occurs at a faster rate, the system is likely to keep more pages of the job in the processor. For example, with fewer physical I/O operations, pages are not paged out while the job waits for those physical I/O operations to occur.

How SMF Recording Changes When You Run BatchPipes

Because BatchPipes eliminates physical I/O for read and write requests, some SMF I/O recording and accounting information does not appear, such as:

- Type 14 and 15 records for the datasets that are now “piped” are not created
- EXCP data in Type 30 and other job-related records has a value of zero.

Automating Operations for BatchPipes

BatchPipes monitors and reports on three exception conditions: jobs that are waiting, idle, or waiting-in-open for longer than an installation-specified threshold value. Each of these conditions suggests possible actions to be performed either by operations or by automated programs. The exception messages are:

```
ASFP401E READER JOB IDLE ON PIPE ...
ASFP403E READER JOB WAITING FOR DATA ON PIPE ...
ASFP405E READER JOB WAITING FOR OPEN ON PIPE ...
ASFP407E WRITER JOB IDLE ON PIPE ...
ASFP409E WRITER JOB WAITING FOR DATA ON PIPE ...
ASFP411E WRITER JOB WAITING FOR OPEN ON PIPE ...
```

Using automated operations, you can try to determine the cause of the wait and take the appropriate actions. For example, message ASFP403E tells the operator that a reader is in a wait threshold exception; that is, the reader has been waiting longer than the inactivity threshold value set for jobs in a waiting state. In the case of this reader, one of the following could have occurred:

- The inactivity threshold value is set too low for the system or for the scheduling characteristics of these jobs. Use the BatchPipes DISPLAY command to check the threshold settings.
- At least one writer job exists but data is not flowing. In this case, the writer is idle — doing work unrelated to BatchPipes.

The appearance of message ASFP403E might trigger an automated operations program to determine the jobname of the corresponding writer job, and use that information to tell the operator of the delay or automatically issue an operator command to improve the writer's performance specification.

For complete message texts of the exception messages, see “ Appendix D: BatchPipes Messages” on page 203. For descriptions of the inactivity thresholds and how to set them, see “Setting Inactivity Thresholds through ASFPBPxx” on page 36.

Checklist of Planning and Implementing Tasks

The following checklist has been created and used by a BatchPipes customer. Tasks listed in the table are intended for both system programmers and application developers. IBM recommends that you use a checklist such as this at your installation.

Figure 12 (Page 1 of 2). BatchPipes Implementation Checklist

Tasks to be performed:	**JOBA**	**JOBB**	**JOBC**	**JOBD**
Analysis				
a. Review current job flow				
b. Develop new job flow				
c. Review suggested implementation				
d. Review impact of job completion change				
e. Obtain approval for time change and job change				
Capacity Review				
a. Determine if sort/work space is adequate for new flow				
b. Determine if tape drive capacity is adequate for new flow				
c. Determine if CPU capacity is adequate for new flow				
d. Determine if initiator availability is adequate				
Program Changes				
a. Determine if 'hardened' file is needed				
b. Modify program to create 'hardened' file				
c. Create Change Request				
JCL Changes				
a. Alter writer job(s) DDs (DSN,DCB, SUBSYS)				
b. Alter reader job(s) DDs (DSN,DCB, SUBSYS)				
c. Review RACF protection of new DSNs				
d. Add 'hardened' file DD to PROC (if required)				
e. Update restart instructions				
f. Add MAIN card to JCL so job will run on SYSx				
g. Coordinate other pending changes for job(s)				
h. Update disaster recovery JCL				
i. Update Change Request				
Job Scheduler Changes				
a. Determine what predecessor/successor hooks need to be changed				
b. Change predecessor/successor hooks				
c. Update Change Request				
d. Update Change Request triggering: (for example, if you have established the creation of a data set as an event trigger for other jobs and you now pipe that data set, you must validate that the event trigger still occurs).				

Figure 12 (Page 2 of 2). BatchPipes Implementation Checklist

Tasks to be performed:	JOBA	JOBB	JOBC	JOBD
Record Auditing Program				
a. Determine what balancing changes are required				
b. Review changes				
c. Implement changes				
d. Update Change Request				
Change Control				
a. Issue change control record				
b. Obtain change control approval				
c. Implement change				
Other				
a. Report results of implementation - before/after				

Chapter 4: Initializing BatchPipes Subsystem

This chapter is for those who initialize BatchPipes based on their installation's workloads, resources, and requirements. The procedures described assume that you have installed BatchPipes according to the program directory provided with the product.

The initial instance of a BatchPipes subsystem requires an IPL and then issuance of a START command. With proper planning, subsequent initializations of BatchPipes subsystems do not require an IPL.

Characteristics of a BatchPipes Subsystem

Each BatchPipes subsystem you initialize runs as a started task in its own non-swappable and pageable address space. The subsystem runs under the job's TCB, except for subsystem management and command processing. The storage that provides for the flow of data from one job to another is managed and owned by the BatchPipes subsystem.

Summary of Initialization Tasks

Initialization tasks and the sections where you can find more information follow; required tasks are identified by an asterisk (*).

- * Define the subsystem to MVS, which includes choosing a name for the subsystem. See “Defining the Subsystem to MVS” on page 30.
- * Create a JCL procedure for starting the subsystem and add it to SYS1.PROCLIB. See the Program Directory for BatchPipes.
- * Authorize the BatchPipes code through the authorized program facility (APF). See “Authorizing the BatchPipes Code through Authorized Program Facility (APF)” on page 31.
- * Ensure that enough common storage area (CSA) exists for BatchPipes. See “Reserving CSA for BatchPipes” on page 31.
- * Ensure that each BatchPipes subsystem has a system linkage index (LX) for its use. See “Evaluating the Use of Linkage Indexes” on page 31.
- Activate SMF recording for BatchPipes. See “Activating SMF Recording” on page 32.
- Set a subsystem recording interval to allow SMF to give periodic reports. See “Setting a Subsystem Recording Interval” on page 32.
- Evaluate the need to change system resources manager (SRM) parameters or define workload management goals. See “Performance Tuning for BatchPipes” on page 32.
- Restrict the use of BatchPipes through RACF*, or an equivalent product. See “Controlling the Use of BatchPipes” on page 34.
- Define inactivity thresholds in ASFPBPxx member of SYS1.PARMLIB. See “Setting Inactivity Thresholds through ASFPBPxx” on page 36.
- Make changes for any sort programs that run in the pipeline. See “Making Changes for Sort Jobs” on page 37.

Following the descriptions of initialization tasks is “Example of Initializing a BatchPipes Subsystem” on page 60 that contains an example of each task. For more information about SYS1.PARMLIB parameters, see *MVS/ESA Initialization and Tuning Reference*.

Defining the Subsystem to MVS

The BatchPipes subsystem must be defined on the system that converts and executes the jobs using that subsystem.

Choose a meaningful name for the BatchPipes subsystem; it must:

- Be one to four characters in length
- Consist of numbers or letters in the alphabet
- Begin with a letter
- Not conflict with any other subsystem name in the same system or any system command.

System programmers, application developers, and operators will subsequently use the subsystem name. See “Using the Name of the BatchPipes Subsystem.”

To define the subsystem to MVS, add the name to the IEFSSNxx member of SYS1.PARMLIB. This member contains the names of all subsystems. The name in IEFSSNxx must match the name of the SYS1.PROCLIB member that contains the start procedure for the BatchPipes subsystem. If you have more than one subsystem defined in IEFSSNxx, then each one must correspond to a member name in SYS1.PROCLIB.

You cannot dynamically change the IEFSSNxx member of SYS1.PARMLIB. Therefore, if you know you will be adding another instance of BatchPipes or a later version of BatchPipes, add two or more subsystem names to IEFSSNxx at the original initialization of BatchPipes. By taking this step, you might avoid a reIPL later when you initialize a second BatchPipes subsystem.

If your installation requires the BatchPipes subsystem to run under the **primary** JES **by default**, include “ASFPMSSI” immediately after the subsystem name specification in the IEFSSNxx member of SYS1.PARMLIB. Remember, however, that BatchPipes has no need for a JES job number or for SPOOL space.

Using the Name of the BatchPipes Subsystem

The SUBSYS=*bp-subsysname* parameter on the DD statement tells MVS that the BatchPipes subsystem with that name is to control data movement. For example, in the following JCL, the BatchPipes subsystem named BP01 manages the data movement.

```
//SAMPJCL DD DSN=STEVE.TEMP,SUBSYS=BP01, . . .
```

BatchPipes provides a set of operator commands that control a BatchPipes subsystem and monitor its activities. These commands begin with a command prefix. By default, the command prefix is the name of the BatchPipes subsystem. You can change the command prefix to a character string other than the name of the BatchPipes subsystem. To do this, use the PARM keyword on the IBM-supplied EXEC statement to identify the prefix or use the PREFIX parameter on the START command that starts the BatchPipes subsystem. See “Starting a BatchPipes Subsystem” on page 69 to learn how to code the PREFIX parameter.

The following examples illustrate some of the commands an operator would use to control and monitor a BatchPipes subsystem named BP01:

- An operator can start the subsystem with:

  ```
  START BP01,SUB=MSTR
  ```

  and stop the subsystem with:

  ```
  BP01 STOP
  ```

- The status of the subsystem named BP01 is obtained by the following command:

  ```
  BP01 STATUS
  ```

  As part of message ASFP210I in response, the system might return the following line:

  ```
      #JOBS=2        #OPEN=2      #ALLOC=2       STATUS=ACTIVE
  ```

  which identifies the subsystem as being in the active state, with two jobs that have allocated and opened at least one pipe.

The subsystem name will show up in SMF data and operator displays. Your automation products and monitors can key off of this name.

Authorizing the BatchPipes Code through Authorized Program Facility (APF)

To **IEAAPFxx**, add 'SYS1.SASFPLIB' to the list of program library names of programs that require APF-authorization.

If you are running MVS/ESA SP 5.0 or a later release, you can use dynamic APF services to dynamically add and delete entries from the APF table.

Reserving CSA for BatchPipes

Each BatchPipes subsystem uses virtual storage in the common service area (CSA) - - both above 16 megabytes and below 16 megabytes. Evaluate whether you need to change the setting of the CSA parameter in IEASYSxx member of SYS1.PARMLIB to reserve more CSA.

Plan for each active BatchPipes subsystem to use approximately 3 kilobytes of virtual storage below 16 megabytes and 186 kilobytes above 16 megabytes.

Evaluating the Use of Linkage Indexes

Another step you must take is to evaluate your installation's use of system linkage indexes (LXs). Do you have enough LXs for your BatchPipes subsystems?

On the NSYSLX keyword in **IEASYSxx**, you might need to increase the number of LXs. Assume that you will need one system LX value for each subsystem you are initializing. BatchPipes attempts to reuse its previous LX value if the subsystem is restarted within an IPL. This means that when you restart a subsystem after an operator has issued a CANCEL command, you do not need to increase the number.

Activating SMF Recording

In SMF Type 91 records, BatchPipes provides information about the BatchPipes subsystem, the pipes, and the jobs that use pipes. Activate BatchPipes recording by initializing the **SMFPRMxx** member of SYS1.PARMLIB in one of the following two ways:

- Include SUBSYS(*bp-subsystem-name*, ... TYPE(..91..) ..) in SMFPRMxx, where *bp-subsystem-name* is the name of the BatchPipes subsystem
- Omit SUBSYS(*bp-subsystem-name*, ... TYPE(..91..) ..) and include Type 91 on the SYS parameter. In this case, BatchPipes uses the values set on the SYS parameter that controls system-wide SMF recording.

Setting a Subsystem Recording Interval

Record Type 91 subtypes 2 and 12 occur at an interval that you specify. The interval is known as the **subsystem recording interval**. Activate SMF interval recording in the SMFPRMxx member of SYS1.PARMLIB in one of two ways:

- Include the INTERVAL option specified on the SUBSYS statement. Include SUBSYS(*bp-subsystem-name*, ... INTERVAL(*hhmmss*) ..), where *bp-subsystem-name* is the name of the BatchPipes subsystem and *hhmmss* is the recording interval.
- Omit SUBSYS(*bp-subsystem-name*, ... INTERVAL(*hhmmss*) ..) in SMFPRMxx to request a recording interval at the SYS level. In this case, BatchPipes uses the values set on the SYS parameter that controls system-wide SMF recording.

For more information about SMF Type 91 records, see “SMF Recording for BatchPipes” on page 63. Descriptions of the SMF Records appear in “ Appendix A: SMF Record Descriptions” on page 153.

Performance Tuning for BatchPipes

How you tune your system for BatchPipes depends on whether you are using SRM parameters or using workload management, introduced in MVS/ESA SP Version 5:

- Use SRM parameters if you are using a pre-Version 5 system
- Use workload management if you are running Version 5 and are using workload manager (WLM) goal mode.

In either case, you need to determine:

- How important the pipeline jobs are in relationship to the other work in the system
- What resources the jobs need
- How you can measure the performance of the jobs.

If you want your BatchPipes work to get preferential attention from the system, keep the following goals in mind:

PERFORMANCE GOALS FOR JOBS IN A PIPELINE

- Eliminate or minimize swapping
- Avoid contention among the jobs on the pipeline
- Avoid contention between jobs on the pipeline and other work in the system
- Minimize the amount of time jobs spend waiting on full or empty pipes
- Improve the I/O performance of input to the beginning of the pipeline and output from the end of the pipeline, as well as all physical I/O associated with the pipeline jobs.

Jobs in a pipeline compete with other work in the system for the use of the processor, central storage, and other resources. When the system is very busy, increasing those jobs' use of these resources might have an adverse effect on other work. For this reason, you need to weigh carefully how you tune for performance.

Tuning SRM Parameters

If you are running Version 5 in WLM compatibility mode, you make changes to installation performance specifications (IPS) and the installation control specifications (ICS).

Changing IEAIPSxx Member

Create a separate domain for the jobs in a pipeline, using the DMN keyword, to separate the jobs in a pipeline from other types of work in the system. The jobs within a domain contend as a **set of jobs** for the use of central storage. For each domain:

- Set the minimum and maximum MPL values, using the CNSTR keyword, to limit the number of address spaces allowed in central storage at one time.
  - Set the minimum MPL equal to, or greater than, the number of jobs on the pipeline.
  - Set the maximum MPL equal to, or greater than, the minimum value.
- Set the service level, using the ASRV keyword, so that service is balanced among the jobs within the domain.

Create a new performance group for each domain, using the PGN keyword.

- Set the dispatching priority of the jobs, using the DP keyword. Place all the jobs in a fixed priority; that is, use the DP=Fxy setting.
- Change the length of the performance group period. Because use of processor and processor storage change with BatchPipes, you might need to set the durations, as controlled by the DUR parameter.

Changing the IEAICSxx Member

Create entries to identify the jobs (using parameters such as TRXNAME or TRXCLASS) and associate them with the correct control (PGN parameter) or reporting (RPGN parameter) performance groups. *MVS/ESA Initialization and Tuning Guide* contains a list of attributes available for work classification by subsystem.

Using Workload Management

WLM requires you to define performance goals for the pipeline in a service policy. Then the system decides how much resource, such as processor and storage, should be given to the pipeline jobs to meet the goals.

- Define the service class that represents the BatchPipes jobs.
- Define goals for the service class. Assign a business importance to each goal. Because BatchPipes jobs tend to be long-running, you probably would use the velocity goals. To compare before- and after-BatchPipes performance, compare the velocity fields in RMF reports.
- Using classification rules, assign the BatchPipes jobs to the service class you have defined. Because you define service class by job name, it is important that you have a meaningful naming convention for job names.

For other performance-related information, see “Performance Considerations for Application Developers” on page 114 in the application developers' section.

Controlling the Use of BatchPipes

Through RACF or an equivalent security product, you can control which users and jobs have access to BatchPipes subsystems and specific pipes Reasons you would want this control include the following:

- If you charge for the use of pipes, you do not want anyone except authorized users to run jobs using BatchPipes.
- To control the use of the resources that pipes require, you might want to limit the use of pipes.
- It is possible for a user to, inadvertently or deliberately, submit a job that would join a pipeline that is not intended for its use. The result is that the input to the pipe includes unwanted records or the output has missing records.

  For example, consider JobA that passes data through a pipe to JobB with a `'.'`. Another job could join JobA as a writer and add unwanted input to JobB. One logical occurrence of this scenario could be when two versions of a job exist: one for production and one for testing. If the purpose of the job is to credit money to a bank account, the additional records or lost records could cause big problems.

To use RACF to restrict use of a BatchPipes subsystem or a specific pipe, take the following steps. This list assumes that you have already defined the FACILITY class.

1. Create a DATASET profile for each data set that you plan to define as a pipe.
2. Create profiles in the FACILITY class to restrict access to BatchPipes. The resource name that represents the BatchPipes subsystem is

```
'PSP.ASFPPIPE.bp01'
```

where bp01 is the name of the BatchPipes subsystem that you define in IEFSSNxx member of PARMLIB. Create the profiles through the following RACF command:

```
RDEFINE  FACILITY  PSP.ASFPPIPE.bp01 UACC(NONE)
```

3. Give users the appropriate access authority to the FACILITY profile:

```
PERMIT  PSP.ASFPPIPE.bp01 CLASS(FACILITY)
                   ID(USERID) ACCESS(READ)
```

4. If you have not already done so, activate the FACILITY class.

```
SETROPTS  CLASSACT(FACILITY)
```

5. To protect a pipe, use the same generic protection that you use in the non-BatchPipes environment to protect a data set. If you already have generic resource profiles for data sets, using pipes with either the same names as protected data sets or with the same high-level qualifiers defined in a data set profile, allows you to restrict access to a specific pipename.

Because data set profiles might allow many users access to a set of data sets, you might want to consider creating new profiles for piped sets and create a standard naming convention for pipes to facilitate profile and access list creation.

Controlling the Use of the BatchPipes PIPEDEPTH Parameter

To allow the use of large pipes (pipes with a PipeDepth greater than the value specified on MAXBUFNO parmlib setting), a new facility must be created and users must be given permission to use this facility. The new resource name is:

```
PSP.ASFPPD.BP01
```

where bp01 is the BatchPipes subsystem name. Once created, you may authorize the use of the resource by permitting access to those users who are allowed to specify large values on the PipeDepth parameter. Users must have READ access or greater to specify a values larger than the MAXBUFNO parmlib setting on the PipeDepth parameter.

Here are some implications of using large PipeDepth:

- Consider system paging space when you allow large pipes to exist. When you establish a pipe with pipedepth=30, you are not dramatically changing the number of active pages in use by a job. However, when you allow, for example, pipedepth=15000, you will be changing the number of active pages of storage to a great extent. The system paging will increase, and the need for page space will increase as well.
- Consider the effect on coupling facility usage in a PipePlex. Allowing large pipes will considerably affect storage use in the coupling facility. You may want to consider only allowing the PipeDepth parameter to be used on local mode BatchPipes subsystems.

Controlling the Use of BatchPipeWorks Fittings

Through RACF or an equivalent security product, you can control which users can use BatchPipeWorks fittings. A fitting assumes the authority state of its associated application.

BatchPipeWorks enforces some rules for authorizing fittings:

- If an application runs in authorized state, you must give appropriate access.
- If an application runs in unauthorized state, you do not need to give users access unless you have already defined BatchPipeWorks to RACF.

Then, you must give appropriate access. That is, after you define BatchPipeWorks to RACF, all users must be permitted access.

The resource name that represents unauthorized fittings is:

```
'PSP.ASFPFIT.bp01'
```

where bp01 is the name of the BatchPipes subsystem.

The resource name that represents authorized fittings is:

```
'PSP.ASFPAFIT.bp01'
```

Before you can use fittings with authorized applications, PSP.ASFPAFIT.bp01 must be defined. Otherwise, BatchPipes does not allow the use of fittings on authorized programs and issues message ASFP332I and rejects the OPEN request.

If PSP.ASFPAFIT.bp01 is defined, but the user does not have access to resource, BatchPipes issues message ASFP331I and rejects the OPEN request.

To control access to the fittings, create a resource profile in the FACILITY class as follows:

```
RDEFINE FACILITY PSP.ASFPFIT.bp01 UACC(NONE)
```

or

```
RDEFINE FACILITY PSP.ASFPAFIT.bp01 UACC(NONE)
```

To permit access to this resource profile, enter:

```
PERMIT PSP.ASFPFIT.bp01 CLASS(FACILITY) ID(user) ACCESS(READ)
PERMIT PSP.ASFPAFIT.bp01 CLASS(FACILITY) ID(user) ACCESS(READ)
```

If the FACILITY class is not already active, activate it as follows:

```
SETROPTS CLASSACT(FACILITY)
```

Setting Inactivity Thresholds through ASFPBPxx

BatchPipes reports on the inactivity of jobs in three ways: through the BatchPipes monitor, in SMF Type 91 records, and through threshold messages (ASFP401 through ASFP412). “Inactivity” refers to jobs that are:

- Waiting
- Idle
- Waiting-for-open.

You can set an **inactivity threshold** for each of those types of inactivity. If a job is idle, waiting, or waiting-for-open for longer than the inactivity threshold setting, a BatchPipes message appears on the operator's console. A job that is inactive for a period of time that equals or exceeds the setting is called an **inactivity threshold exception**, or simply a **threshold exception**.

For example, if you set an inactivity threshold of 20 minutes for idle jobs, a message appears when a writer or reader is idle for 20 minutes. That job has a threshold exception.

There are two ways to set inactivity thresholds:

- Through the BatchPipes member ASFPBPxx, as described in this section
- Dynamically, through the BatchPipes SET command, as described in “Setting an Inactivity Threshold through the SET Command” on page 73.

Setting inactivity intervals through ASFPBPxx requires that member ASFPBPxx be in SYS1.PARMLIB. When the BatchPipes subsystem is installed, sample member ASFPBP00 exists in SYS1.SAMPLIB. You can either copy the member from SYS1.SAMPLIB to SYS1.PARMLIB and modify it, or create your own ASFPBPxx member in SYS1.PARMLIB. Then use the following parameters in the ASFPBPxx member:

- WAIT=*nnn* or OFF
- IDLE=*nnn* or OFF
- WAITOPEN=*nnn* or OFF

Where *nnn* is the number of minutes, ranging from 0 to 300, and OFF means that BatchPipes does not monitor for that particular type of threshold exception. Default inactivity threshold values are 15 minutes.

Activate member ASFPBPxx at subsystem start through the MEM parameter on the START command; see “Starting a BatchPipes Subsystem” on page 69. Or, you can use the MEMBER parameter on the BatchPipes SET command; see “Setting an Inactivity Threshold through the SET Command” on page 73.

Making Changes for Sort Jobs

If possible, avoid having a sort program repeatedly allocating sort work space. If your sort program accepts a file size option, you can define work space before it starts the sort operation.

Generally, BatchPipes lowers the use of DASD storage for temporary data sets. However, DASD storage for sort work space might go up. And, if you have two sorts running in the pipeline at the same time, both sorts need work space at the same time.

Creating a Coupling Facility Structure for the Pipeplex

To use cross-system piping, you must define a coupling facility list structure in your installation's Coupling Facility Resource Management (CFRM) policy. Specify the name of the structure and the amount of storage that is needed for the structure.

As an additional user of the coupling facility, the pipeplex can have performance impact on the other functions that use the coupling facility, such as DB2 and IMS.

For general help in defining a structure, see the appendix on the Administrative Data Utility in OS/390 MVS Setting Up a Sysplex.

Besides creating the pipeplex structure, you will also need to know how to provide recovery if the structure or a connection to it fails, and how to apply periodic maintenance to the coupling facility.

This section explains these considerations in detail.

Naming the Pipeplex CF Structure

Name the structure SYSASFPssnm, where ssnm is the name associated with the BatchPipes subsystems within the pipeplex. For guidelines on naming MVS subsystems, see *OS/390 MVS Using the Subsystem Interface*.

Defining Storage for the Pipeplex CF Structure

When you define SYSASFPssnm in the CFRM policy, specify the maximum size of the coupling facility structure in 1K blocks, using the SIZE(size) parameter of the Administration Data utility.

IBM provides you with the ASFPECAL REXX exec to help you determine the size of the coupling facility structure. The exec is provided as member ASFPECAL of data set SASFPSAM.

You will need to include the following information on the command that you use to run ASFPECAL:

PIPES(nnn) The maximum number of cross-system pipes that are to be in use at one time. Choose a value that is high enough to allow for future growth in the pipeplex.

SYSTEMS(nn)
Maximum number of BatchPipes subsystems that are to be run in the pipeplex at any one time (1 subsystem per MVS system in the pipeplex). This value must be 2 or more

BUFFERS(nnn)
Average number of buffers for each cross-system pipe within the pipeplex.

The ASFPECAL exec uses this information to calculate the size of the pipeplex SYSASFPssnm structure.

Assume, for example, that you plan to use the following pipeplex configuration at your installation:

- Up to 100 cross-system pipes active at one time.
- Up to 4 BatchPipes subsystems active in the pipeplex at one time.
- An average of 20 buffers for each cross-system pipe in the pipeplex.

To find out the maximum structure size to define, you would enter the following command on the TSO/E command line:

```
EX 'SYS1.SASFPSAM(ASFPECAL)' 'pipes(100) systems(4) buffers(20)'
```

The ASFPECAL exec displays the following message to your terminal:

```
You have entered the following values...
  The number of pipes:    100
  The number of systems:  4
  The number of buffers:  20
  The estimated size of your structure is: 71740  1K blocks.
```

Based on this example, you would specify the maximum structure size in the CFRM policy as 71740 1K blocks.

IBM recommends that you begin your pipeplex configuration with the following initial values:

- 100 pipes
- Current number of MVS systems
- 20 buffers per pipe.

The following are examples of other possible pipeplex configurations, based on results from the ASFPECAL exec.

- Sample input: pipes(200) systems(4) buffers(30)

  ASFPECAL results are:

```
You have entered the following values...
  The number of pipes:    200
  The number of systems:  4
  The number of buffers:  30
  The estimated size of your structure is: 203736  1K blocks.
```

- Sample input: pipes(200) systems(4) buffers(20)

  ASFPECAL results are:

```
You have entered the following values...
  The number of pipes:    200
  The number of systems:  4
  The number of buffers:  20
  The estimated size of your structure is: 139432  1K blocks
```

- Sample input: pipes(100) systems(4) buffers(20)

  ASFPECAL results are:

```
You have entered the following values...
  The number of pipes:    100
  The number of systems:  4
  The number of buffers:  20
  The estimated size of your structure is: 71740  1K blocks.
```

- Sample input: pipes(100) systems(8) buffers(20)

  ASFPECAL results are:

```
You have entered the following values...
  The number of pipes:    100
  The number of systems:  8
  The number of buffers:  20
  The estimated size of your structure is: 73796  1K blocks.
```

Modifying the Size of the Pipeplex CF Structure

If necessary, you can manually rebuild the pipeplex coupling facility structure through the SET XCF command. This capability gives your installation the flexibility to modify the size of the pipeplex structure without having to reIPL the sysplex.

You might rebuild the pipeplex structure, for example, to accommodate an increase in cross-system piping at your installation. Or, you might decrease the size of the pipeplex structure if you later determine that its initial size is larger than necessary.

Loss of a Coupling Facility Structure

If a BatchPipes subsystem running in a pipeplex detects a structure failure, or loses connection to the coupling facility structure, the subsystem cannot service the jobs using cross-systems pipes. If the pipeplex structure fails, or a subsystem's connection to the structure fails, you cannot rebuild the pipeplex coupling facility structure as you might expect.

As a result, the subsystem issues the following action message:

ASFP429E BATCHPIPES ssnm CROSS-SYSTEM PIPES HALTED ON sysname {A BATCHPIPES CF STRUCTURE FAILURE OCCURRED.] LOST CONNECTIVITY TO THE BATCHPIPES CF STRUCTURE.}

A job that attempts to read from a cross-system pipe that is managed by the failing subsystem, or write to it, receives a X'BC6' system abend and fails with a physical I/O error.

A job that attempts to open or close a cross-system pipe that is managed by the failing subsystem fails, and one of the following error messages is written to the job log:

ASFP339I BATCHPIPES OPEN FAILURE. A CATASTROPHIC SUBSYSTEM ERROR OCCURRED. THE SUBSYSTEM IS NO LONGER ABLE TO OPEN PIPES. JOB=jobname STEP=jobstep DD=ddname SUBSYS=ssname

ASFP342I BATCHPIPES CLOSE FAILURE. A CATASTROPHIC SUBSYSTEM ERROR OCCURRED. THE SUBSYSTEM IS NO LONGER ABLE TO CLOSE PIPES. JOB=jobname STEP=jobstep DD=ddname SUBSYS=ssname

ASFP345I ALLOCATION FAILURE. A CATASTROPHIC SUBSYSTEM ERROR OCCURRED. THE SUBSYSTEM IS NO LONGER ABLE TO CREATE PIPES. JOB=jobname STEP=jobstep DD=ddname SUBSYS=ssname

In response to the loss of the coupling facility structure, the BatchPipes subsystem cancels itself after it has taken whatever action it can to lessen the impact of the failure.

When you have corrected the problem, you can restart the subsystem and rerun the jobs that failed, and any other jobs in the pipeline that might have been affected by the failure.

Performance Impact to Coupling Facility

As an added user of the coupling facility, BatchPipes can affect the performance of the other users of the coupling facility, such as DB2 and IMS. And, the reverse is true: existing users of the coupling facility can affect the performance of the pipeplex.

You can minimize the impact of the pipeplex on other users of the coupling facility by limiting the use of cross-system piping to only jobs that to run on multiple systems.

Error Propagation for Member Subsystems

In a pipeplex, the scope of error propagation is sysplex-wide. Error propagation includes MVS systems, pipeplex subsystems, and the jobs running under the pipeplex subsystems. If a system or a pipeplex subsystem fails, the pipeplex includes error propagation for the pipeplex subsystem as part of clean-up processing. Specifically, the pipeplex takes the following actions to propagate errors:

- Checks the connections of each cross-system pipe to which the failing subsystem was connected and deletes the connections for the failing subsystem's jobs.
- For a partner job that resides on a different MVS system, the pipeplex propagates the I/O error.
- For a partner job that resides on the MVS system where the failing subsystem was running, no propagation occurs. If this job attempt to access the cross-system pipe, however, it receives an I/O error indicating that the subsystem is no longer active.

For example, if subsystem BP01 on MVS system SYS01 fails, an I/O error will be propagated for JOB A on SYS01 to JOB B on SYS02 and JOB C on SYS03.

When a pipeplex subsystem fails, partner jobs on other MVS systems in the pipeplex will receive multiple I/O error propagations from the failing subsystem's jobs.

When multiple I/O error propagations are sent to a partner job, some error propagation messages might not be recorded in the joblog before the partner job ends. This condition can occur when the partner job specifies EROPT=ABE for its pipe data sets, which causes the job to abend on receiving the first I/O error propagation.

Chapter 5: Diagnosing Problems Associated with BatchPipes

This chapter describes some problems your installation might encounter when you run BatchPipes and suggests actions you can take. It also tells you how to gather information that the IBM support center might require in case you turn to them for help.

Handling Problems in a BatchPipes Environment

What signals problems with jobs using BatchPipes? As with non-BatchPipes jobs, the first sign of problems in a pipeline is that jobs stop running. If jobs in a pipeline stop, first verify the relationships among the jobs in the pipeline; use the BatchPipes STATUS command with the FLOW parameter or the BatchPipes monitor. See "Using the BatchPipes STATUS Command" on page 83 or "Using the BatchPipes Monitor" on page 96.

How do you respond to problems in a BatchPipes environment? If either the jobs or the BatchPipes subsystem encounter errors, take one or more of the following actions to collect information:

- Use the STATUS command or the BatchPipes monitor to determine that a job is not hung up waiting for some event, such as a tape mount or the completion of another job.
- Check SYSLOG for any messages pertaining to job abends or BatchPipes subsystem abends.
- Check JOBLOG for any nonzero return code messages.
- Check recent BatchPipes messages. See " Appendix D: BatchPipes Messages" on page 203.
- Verify that DD statements for jobs using pipes (DCB information, DSNAME parameter, SUBSYS parameter) are accurate. Correct specifications are in "Chapter 12: Making Changes to JCL or Dynamic Allocation" on page 117.
- Capture any dumps that result from an abend of a BatchPipes subsystem. Such errors might cause an ABEND `X'BC6'` with a specific reason code. These reason codes are documented in " Appendix E: BatchPipes Codes" on page 283.
- Use the BatchPipes DUMP command to capture data that IBM personnel need, in case you eventually ask for their help. See "Using the BatchPipes DUMP Command" on page 47.

If necessary, you can shut down a BatchPipes subsystem with the BatchPipes STOP or CANCEL commands and restart it without a system IPL; in this case:

- Take a dump
- Cancel any outstanding jobs
- Stop the BatchPipes subsystem
- If jobs still remain, issue the CANCEL command again.

Then you can restart the subsystem and the jobs. If you need to turn to IBM's support center for help, see "Requesting Diagnostic Help from IBM" on page 45 to learn what information they might request.

Taking Dumps at Abnormal Termination

Errors in BatchPipes code may cause an abend X'BC6' with a specific reason code. If an abend occurs and a dump is produced, check the code in " Appendix E: BatchPipes Codes" on page 283. Also, several abend X'001' dumps can occur as a result of I/O errors that are propagated by the initial abend. **IBM recommends** that you capture the initial X'BC6' dump.

Responding to a Job that is Hung

Job hangs that occur while using BatchPipes often result from any situation where only reader(s) or only writer(s) are connected to a pipe. Use the BatchPipes STATUS command or the BatchPipes monitor to determine which jobs are waiting, the length of time of the delay, and whether these jobs are writers or readers. Below are scenarios that would lead to job hangs and the responses you can make:

- A job may be waiting for another job to open (connect to a pipe), but the other job may have abended before it connected to the pipe.

  Response: Resubmit the job that did not connect to the pipe, or cancel both jobs and resubmit the set of jobs.

- Not enough initiators are available for all the jobs in the pipeline.

  Response: Start more initiators or change job classes of already-started initiators.

- A job may be in wait-in-open state in a many-to-one, one-to-many, or many-to-many pipeline situation. If the job arrived "late" to the pipe, it is possible that the other jobs connected to the pipe finished processing and closed the pipe before the late job issuing an open to the pipe.

  Response: Make sure no job tries to connect to a pipe after other writers or readers have finished using the pipe. In a many-to-one or one-to-many pipe, the jobs on the "many" side may be scheduled before the job on the "one" side.

  Another response might be to make sure enough initiators exist for the cloned jobs.

- If the DSNAME that identifies the pipe name in the DD JCL statement is not the same for both writer and reader jobs, the jobs hang as a result of connecting to different pipes.

  Response: Check the JCL of any jobs that are hanging or issue the STATUS command with the FLOW parameter. Look for the specific pipe name.

- If the SUBSYS parameter on the DD JCL statement is not the same for both writer and reader jobs, and the subsystems specified actually exist (and are started), jobs may hang trying to connect to pipes in different subsystems.

  Response: Check the JCL of any jobs that are hanging.

- Jobs specifying temporary dataset names as pipe names on the DD JCL statement will hang.

  Because the system creates a unique name for each temporary DSNAME specified, the writer and reader jobs connect to different pipes, and both jobs will wait indefinitely for the other to connect. The waiting jobs will be swapped out permanently and hang as a result.

Response: Do not use any temporary (ampersand-qualified) DSNAMEs to specify pipe names.

- If a BatchPipes STOP command is issued while a job is waiting for another job to open (connect to a pipe), the job that is connected to the pipe will hang. Any existing processing in the pipeline will complete, however new jobs will not be allowed to initiate connections to a pipe.

  Response: Cancel any jobs that are left hanging after a BatchPipes STOP has been issued, or cancel the BatchPipes subsystem.

If you plan to contact IBM's support center for help in diagnosing a job hang situation, issue the BatchPipes DUMP command to capture information at time of error before you cancel and restart the jobs and subsystem. See “Using the BatchPipes DUMP Command” on page 47.

Requesting Diagnostic Help from IBM

Sometimes the problem you encounter with your jobs running BatchPipes requires help from IBM. When you encounter a job abend and are unable to diagnose the problem, call on IBM's support center. They might ask for information that would require that you:

- Trace events in the BatchPipes subsystem by using the BatchPipes TRACE command. See “Using the BatchPipes TRACE Command” on page 46 for more information.
- Take a dump of BatchPipes information. See “Using the BatchPipes DUMP Command” on page 47 for more information.

Other information they might request is:

- Output of the STATUS command
- The job's JCL
- DCB information in the application code
- Whether the jobs are user applications or vendor products; if vendor products, then which ones
- Abend codes
- Error messages
- Joblog messages
- Console displays at time of error, such as:
  - STATUS,FLOW,J=job in the pipeline
  - STATUS for the subsystem
- BatchPipes monitor output at time of error
- SYS1.LOGREC entries around time of error.

Using the BatchPipes TRACE Command

BatchPipes provides trace information that IBM can use to follow the processing of a BatchPipes subsystem. **Do not run with BatchPipes TRACE turned on** unless you are attempting to recreate a problem. By default, BatchPipes runs with the tracing capability turned off.

Turn BatchPipes TRACE on through the BatchPipes TRACE command. The syntax of the TRACE command is as follows:

```
┌─── BATCHPIPES TRACE ──────────────────────────────────────────────────┐
│                                                                       │
│ ►►── bp-cmd-prefix───►──TRACE────────────┬─,ALL──────┬───────────────►◄ │
│                                          ├─,FLOW─────┤                │
│                                          ├─,FUNCTION─┤                │
│                                          ├─,OFF──────┤                │
│                                          └─,STATUS───┘                │
│                                                                       │
└───────────────────────────────────────────────────────────────────────┘
```

Parameters on the BatchPipes TRACE command are:

FLOW
Traces the flow of control, giving module entry and module exit. Sets FUNCTION trace off.

FUNCTION
Traces BatchPipes function started and function completed. Sets FLOW trace off.

ALL
Sets both FLOW and FUNCTION tracing on.

In response to FLOW, FUNCTION, or ALL, BatchPipes issues the following message:

```
ASFP212I BATCHPIPES BP01 TRACING STARTED
```

When you have finished recreating the problem, use the following command to stop TRACE FLOW and TRACE FUNCTION.

bp-cmd-prefix TRACE,OFF

In response to this command, BatchPipes issues the following message:

```
ASFP212I BATCHPIPES BP01 TRACING STOPPED
```

To find out whether BatchPipes TRACE is active issue the command:

bp-cmd-prefix TRACE,STATUS

In response to this command, BatchPipes issues ASFP213I:

```
ASFP213I BATCHPIPES BP01 TRACE STATUS: INACTIVE
                                       ALL
                                       FLOW
                                       FUNCTION
```

Using the BatchPipes DUMP Command

The BatchPipes DUMP command produces an unformatted dump that is intended to be used by IBM's support center. Data in the dump includes the BatchPipes subsystem address space and BatchPipes-related information associated with pipes and jobs using pipes.

Use the DUMP command if you suspect an error in the BatchPipes subsystem and the system did not automatically produce an ABEND dump. One example when you would use the DUMP command is in the case of hung jobs.

IBM recommends that you use the BatchPipes DUMP command for BatchPipes jobs, rather than the MVS DUMP command.

Use parameters on the DUMP command to limit the scope of the dump to one or more specific jobs (JOB parameter) or one or more specific pipes (PIPE parameter). Figure 13 summarizes the contents of the dumps that result from BatchPipes DUMP commands:

Figure 13. Using the BatchPipes DUMP Command

If you use:	**The dump contains:**
DUMP with no parameters	BatchPipes subsystem and information pertaining to all jobs and pipes using the subsystem
DUMP with JOB or PIPE parameters or both	BatchPipes subsystem, information pertaining to all specified jobs and pipes using the subsystem, and up to 14 unique address spaces of all jobs specified.

Because the amount of data in the dump can be very large, limit the scope of the DUMP command as much as possible. For example, if you can isolate the problem to one or two jobs or pipes, specify those jobs or pipes when you issue the DUMP command. The BatchPipes STATUS command or the BatchPipes monitor might help you isolate the source of the problem.

The syntax of the DUMP command is as follows:

```
┌── BATCHPIPES DUMP ───────────────────────────────────────────────────────────────┐
│                                                                                  │
│ ►►── bp-cmd-prefix───►──DUMP──────►                                              │
│                                                                                  │
├►──┬──┬─────────────────────────┬──┬───┬──┬───────────────────────────┬──┬──►◄
│   │  └─┬─,J=───┬─┬─jname──┬───┘  │   │  └─┬─,P=─────┬─┬─pname──┬───┘  │
│   │    └─,JOB=─┘ ├─jname*─┤      │   │    └─,PIPE=──┘ ├─pname*─┤      │
│   │              └─*──────┘      │   │                └─*──────┘      │
│   └─┬─,J=───┬─(─┬─┬─jname──┬─┬─)─┘   └─┬─,P=────┬─(─┬─┬─pname──┬─┬─)─┘
│     └─,JOB=─┘   ▲ ├─jname*─┤ │         └─,PIPE=─┘   ▲ ├─pname*─┤ │
│                 │ └─*──────┘ │                      │ └─*──────┘ │
│                 │        (1) │                      │        (2) │
│                 └─────,──────┘                      └─────,──────┘
│                                                                                  │
└──────────────────────────────────────────────────────────────────────────────────┘
```

Notes:

1. *jname* can be repeated up to 9 times
2. *pname* can be repeated up to 9 times.

Parameters on the BatchPipes DUMP command are:

JOB=jname|jname*|*
J=jname|jname*|*

One or more job names. If you specify more than one name, separate them with commas and enclose them with parentheses. Use an asterisk to indicate wild card. A single asterisk requests a dump of information associated with all BatchPipes jobs; information might include up to 14 unique address spaces associated with the jobs.

PIPE=pname|pname*|*
P=pname|pname*|*

One or more pipe names. If you specify more than one name, separate them with commas and enclose them with parentheses. Use an asterisk to indicate wild card. A single asterisk requests a dump of information associated with all pipes; information might include up to 14 unique address spaces associated with the pipes.

An Example of the BatchPipes DUMP Command

To dump information associated with BP01 and capture information pertaining to all jobs whose names begin with BPJOB or BPDD*, issue:

```
BP01 DUMP,JOB=(BPJOB*,BPDD*)
```

The resulting information includes:

- The BP01 subsystem address space
- BatchPipes information about all jobs that are using the BP01 subsystem
- BatchPipes information about all pipes controlled by the BP01 subsystem
- Up to 14 unique address spaces of jobs whose names begin with “BPJOB” and “BPDD.”

Chapter 6: Managing the BatchPipes Subsystem

This chapter is for those who initialize the BatchPipes for OS/390 based on their installation's workloads, resources, and requirements. The procedures that follow assume that you have already installed an initial instance of BatchPipes.

Subsequent BatchPipes subsystems are easier to initialize because you do not have to repeat all the steps that were done for the first. There are however, some things to consider:

- What is the name of the second subsystem? Specify that in IEFSSNxx member.
- Do SMF subsystem recording intervals change?
- Are there new RACF considerations?
- Do you need to establish a new set of libraries?

The sections that follow describe such considerations in more detail.

Characteristics of a BatchPipes Subsystem

Each BatchPipes subsystem you initialize runs as a started task in its own non-swappable and pageable address space. The subsystem runs under the job's TCB, except for subsystem management and command processing. The storage that provides for the flow of data from one job to another is managed and owned by the BatchPipes subsystem.

Summary of Initialization Tasks

Initialization tasks and the sections where you can find more information follow:

- Activate SMF recording for BatchPipes. See “SMF Recording for BatchPipes” on page 63.
- Set a subsystem recording interval to allow SMF to give periodic reports. See “Setting a Subsystem Recording Interval for BatchPipes” on page 65page.
- Change the LOGON procedures for users of the BatchPipes monitor. See “Chapter 9: Monitoring BatchPipes” on page 83page.
- Initialize subsystem parameters in the BatchPipes member of PARMLIB. See “Summary of Initialization Tasks”page.

Following the descriptions of initialization tasks is “Example of Initializing a BatchPipes Subsystem” on page 60page,ontains an example of each task. For more information about PARMLIB parameters, See *OS/390 MVS Initialization and Tuning Reference*.

Starting the BatchPipes Subsystem

BatchPipes provides a set of operator commands that control a BatchPipes subsystem and monitor its activities. These commands begin with a command prefix. By default, the command prefix is the name of the BatchPipes subsystem. You can change the command prefix to a character string other than the name of the BatchPipes subsystem. To do this, use the PARM keyword on the IBM-supplied EXEC statement to identify the prefix or use the PREFIX parameter on the START command that starts the BatchPipes subsystem. See "Starting a BatchPipes Subsystem" on page 69page for an example of how to code the PREFIX parameter on the BatchPipes START command.

The following examples show some of the commands an operator would use to control and monitor a BatchPipes subsystem named BP01:

- An operator can start the subsystem with:

  ```
  START BP01,SUB=MSTR
  ```

  and stop the subsystem with:

  ```
  BP01 STOP
  ```

- The status of the subsystem named BP01 is obtained by the following command:

  ```
  BP01 STATUS
  ```

  As part of message ASFP210I in response, the system might return the following line:

  ```
  #JOBS=2        #OPEN=2      #ALLOC=2       STATUS=ACTIVE
  ```

  which identifies the subsystem as being in the active state, with two jobs that have allocated and opened at least one pipe.

The subsystem name will show up in SMF data and operator displays. Your automation products and monitors can key off this name.

Initializing the ASFPBPxx Member of PARMLIB

Through ASFPBPxx, the BatchPipes parmlib member, you can initialize certain settings for the BatchPipes subsystem, as described in Figure 14. Capitalization of the parameters indicates the shorthand way of specifying them.

Figure 14 (Page 1 of 2). Settings in the ASFPBPxx Member of PARMLIB

To specify:	Use these parameters:	As described on:

The inactivity threshold values (See note.)	Wait=nnnn or OFF
	Idle=nnnn or OFF
	WaitAlloc=nnnn or OFF
	WaitOpen=nnnn or OFF
	WaitClose=nnnn or OFF
	WaitEOF=nnnn or OFF
	WaitTerm=nnnn or OFF
		"Specifying an Alternate Parmlib Data Set for BatchPipes" on page 51page

Whether readers abend if they close the pipe data set before receiving EOF notification	EOFRequired=YES or NO	"Using the EOF Required Parameter" on page 58page

Figure 14 (Page 2 of 2). Settings in the ASFPBPxx Member of PARMLIB

To specify:	Use these parameters:	As described on:
The maximum BUFNO value	MAXBUFNO=nnn	“Setting the Maximum BUFNO Value” on page 58page
The default BUFNO value	DEFBUFNO=nnn	“Setting the Default BUFNO Value” on page 58page
The DB2 system that will be used by BatchPipeWorks for SQL commands	DB2SSID=ssnm	“BatchPipeWorks DB2 Settings” on page 58page
The DB2 plan that will beused by BatchPipeWorks for DB2 access	DB2PLAN=planname	“BatchPipeWorks DB2 Settings” on page 58page
Note: A specific job can override the subsystem inactivity settings by using pipe-level threshold settings. Pipe-level threshold settings are specified on the SUBSYS parameter. See “Setting the Inactivity Thresholds” on page 54page information on setting inactivity thresholds.		

Most likely, you will place ASFPBPxx in PARMLIB. When the BatchPipes subsystem is installed, sample member ASFPBP00 exists in SYS1.SAMPLIB. You can either copy the member from SYS1.SAMPLIB to PARMLIB and modify it, or create your own ASFPBPxx member in PARMLIB.

Activate member ASFPBPxx at subsystem start through the MEM parameter the START command; see “Starting a BatchPipes Subsystem” on page 69page Or, you can use the MEMBER parameter on the BatchPipes SET command; see “Cancelling a BatchPipes Subsystem” on page 71page.

The inactivity threshold parameters at the subsystem level and the other subsystem parameters can be dynamically set through the BatchPipes SET command.

Specifying an Alternate Parmlib Data Set for BatchPipes

BatchPipes supports Logical Parmlib; therefore, you can place BatchPipes parmlib members in any data set in the logical parmlib. You can define up to 10 additional parmlib data sets in a single concatenation that the system uses as a logical parmlib. SYS1.PARMLIB will be concatenated to the end of the logical parmlib sets. An MVS operator can always dynamically change the parmlib data set list with the SETLOAD command, which is described in OS/390 MVS System Commands. You can also modify parmlib settings by using one of the of the following methods:

- Allow the BatchPipes subsystem to initialize with the default parmlib settings. Then, use the BatchPipes SET command to modify the BatchPipes parmlib settings.
- Specify an alternative parmlib data set in the started procedure that you use to start the BatchPipes subsystem. Specify the name of the data set on the IEFPARM DD parameter of the //PROC statement. Then, restart the BatchPipes subsystem to use the alternative parmlib.

Requesting Inactivity Notification

To help you monitor the activities of the writers and readers, BatchPipes reports on the particular states of writers or readers on BatchPipes monitor panels, STATUS command displays, and through the BatchPipes query service.

A writer or reader can be in any of the following states:

- Wait-for-allocation
- Wait-for-open
- Idle
- Wait
- Wait-for-close
- Wait-for-EOF
- Wait-for-termination

Also, BatchPipes allows you to specify **inactivity thresholds** that specify the length of time a writer or reader can stay in one state before BatchPipes sends an inactivity threshold message to the operator. For example, when a job is idle for the length of time equal to the inactivity threshold for idle writers or readers, a message appears on the operator's console. A job that is inactive for a period that equals or exceeds the setting is called an **inactivity threshold exception**. If you set an inactivity threshold of 20 minutes for idle jobs, a message appears when a writer or reader is idle for 20 minutes. That job is an inactivity threshold exception. See “Interpreting Inactivity Threshold Messages” on page 56 page to learn how to interpret the inactivity threshold messages.

Set threshold values low enough to provide helpful information, but high enough to avoid unnecessary messages. Setting a low value for the idle or wait thresholds might result in more messages than you want to receive.

Figure 15 describes the states and identifies the parameters on ASFPBPxx (and also on the SET command) that set the inactivity thresholds. Capitalization of the parameters indicates the shorthand way of specifying them.

Figure 15 (Page 1 of 3). Setting the Inactivity Threshold Values

State	Meaning of the state	Parameter
Wait-for-allocation	A job is waiting for a partner to allocate the pipe data set. Allocation processing does not complete until all writer/reader partners have issued the allocation request (unless you specify the ALLOCNOW subparameter). This state can exist when connections specify the AllocSync subsystem parameter.	WaitAllo

Figure 15 (Page 2 of 3). Setting the Inactivity Threshold Values

State	**Meaning of the state**	**Parameter**
Wait-for-open	A job is waiting for a partner to open the pipe data set. BatchPipes does not build a pipe connection until writer/reader partners are both present (unless you specify the OPENNOW subparameter on the JCL SUBSYS parameter). or All jobs have specified the OPENSYNC subparameter on the JCL SUBSYS parameter to synchronize the opening of the pipe, but the specified number of writers and readers have not yet opened the pipe data set. Those that have opened the pipe are in a wait-for-open state. In monitor panels, these wait-for-open jobs are sometimes called **OpenSync-wait** jobs, as in Figure 35 on page 98	WaitOpen
Idle	A job is performing actions that do not require the services of the BatchPipes subsystem.	Idle
Wait	A job attempts to write to pipe, and is made to wait because the pipe is full. Or, the job tries to read from a pipe and is made to wait because the pipe is empty.	Wait
Wait-for-close	A job is waiting for all writers and readers with pipe connections to close the pipe data set before BatchPipes deletes pipe connections and allows all writers and readers to continue processing. This state requires that all jobs have specified the CLOSESYNC subparameter on JCL SUBSYS to synchronize closure of the pipe.	WaitClose

Figure 15 (Page 3 of 3). Setting the Inactivity Threshold Values

State	Meaning of the state	Parameter
Wait-for-EOF	A reader is waiting at an empty pipe for an EOF indication. All writers have closed the pipe data set and the last writer to close the pipe data set requested (on NOEOF subparameter on JCL SUBSYS) that BatchPipes not send an EOF indication.	WaitEOF
Wait-for-termination	A job is waiting for a partner to arrive at termination. Termination processing does not complete until all writer/reader partners have arrived at termination. This state can occur when connections on a pipe specify the TermSync parameter.	WaitTerm

To check the current settings of inactivity thresholds, use the DISPLAY command. For an example of the resulting message, see Figure 18 on page 76. For an example of a monitor panel that shows the current inactivity of writers or readers using a particular pipe, see Figure 19 on page 77.

If, later, you do not want BatchPipes to track the inactivity thresholds, set THRESH=OFF on the SET command.

Setting the Inactivity Thresholds

The inactivity threshold values in the ASFPBPxx member of PARMLIB apply to all jobs using a BatchPipes subsystem. However, individual jobs can override the subsystem inactivity settings by using pipe-level threshold settings. Pipe-level threshold settings are specified on the SUBSYS parameter, which are described in SmartBatch User's Guide. The allowed keywords, values, and meanings specified for specific pipe-level inactivity thresholds are essentially the same as those specified in PARMLIB, except that they only affect the pipe using pipe-level inactivity settings.

The following parameters set the inactivity threshold values for writers and readers in wait, idle, wait-for-open, wait-for-close, and wait-for-EOF states.

Notes on setting the values of inactivity threshold values:

- A value of 1440 means 24 hours
- A value of 0 equals zero, not OFF.

WAIT=*nnnn*
or OFF

Defines the amount of time a writer or reader can remain in a wait state before BatchPipes issues an inactivity threshold message. Specify WAIT=nnnn where nnnn indicates the number of minutes, from 0 to 1440. If you specify WAIT=OFF, BatchPipes does not the wait state.

The default setting for WAIT=nnnn is the setting established in ASFPBPxx parm lib member. The value specify for the WAIT=nnnn option overrides the setting of the ASFPBPxx PARM LIB member.

IBM recommends that you do not set a value of 0 for the wait threshold.

IDLE=*nnnn*
or OFF

Defines the amount of time a writer or reader can remain idle before BatchPipes sends an inactivity threshold message. Specify IDLE=nnnn, where nnnn indicates the number of minutes, ranging from 0 to 1440. If you specify IDLE=OFF, BatchPipes does not monitor the idle state.

The default setting for IDLE=nnnn is the setting established in ASFPBPxx parm lib member. The value specify for the IDLE=nnnn option overrides the setting of the ASFPBPxx PARM LIB member.

IBM recommends that you do not set a value of 0 for the idle threshold.

WAITALLOC=*nnnn*
or OFF

Defines the amount of time a writer or reader can remain in a wait-for-allocation state before BatchPipes sends a message. WAITALLOC=nnnn indicates the number of minutes, ranging from 0 to 1440.

The default setting for WAITALLOC==nnnn is the setting established in ASFPBPxx parm lib member. The value specify for the WAITALLOC=nnnn option overrides the setting of the ASFPBPxx PARM LIB member.

If you specify WAITALLOC=OFF, BatchPipes does not monitor the wait-for-allocation state.

WAITOPEN=*nnnn*
or OFF

Defines the amount of time a writer or reader can remain waiting-for-open state before BatchPipes sends an inactivity threshold message. Specify WAITOPEN=nnnn, where nnnn indicates the number of minutes, ranging from 0 to 1440. If you specify WAITOPEN=OFF, BatchPipes does not monitor the wait-for-open state.

The default setting for WAITOPEN=nnnn is the setting established in ASFPBPxx parm lib member. The value specify for the WAITOPEN=nnnn option overrides the setting of the ASFPBPxx PARM LIB member.

WAITCLOSE=*nnnn*
or OFF

Defines the amount of time a writer or reader can remain waiting-for-close state before BatchPipes sends an inactivity threshold message. Specify WAITCLOSE=nnnn, where nnnn indicates the number of minutes, ranging from 0 to 1440.

The default setting for WAITCLOSE=nnnn is the setting established in ASFPBPxx parm lib member. The value specify for the WAITCLOSE=nnnn option overrides the setting of the ASFPBPxx PARM LIB member.

If you specify WAITCLOSE=OFF, BatchPipes does not monitor the wait-for-close state.

WAITOF=*nnnn*
or OFF

Defines the amount of time a reader can remain in a wait-for-EOF state before BatchPipes issues an inactivity threshold message. Specify WAITEOF=nnnn, where nnnn indicates the number of minutes, ranging from 0 to 1440. If you specify WAITEOF=OFF, BatchPipes does not monitor the wait-for-EOF state.

The default setting for WAITOF=nnnn is the setting established in ASFPBPxx parm lib member. The value specify for the WAITOF=nnnn option overrides the setting of the ASFPBPxx PARM LIB member.

WAITTERM=*nnnn*
or OFF

Defines the amount of time a writer or reader can remain in a wait-for-termination state before BatchPipes sends a message. WAITTERM=nnnn indicates the number of minutes, ranging from 0 to 1440.

The default setting for WAITTERM=nnnn is the setting established in ASFPBPxx parm lib member. The value specify for the WAITTERM=nnnn option overrides the setting of the ASFPBPxx PARM LIB member.

If you specify WAITTERM=OFF, BatchPipes does not monitor the wait-for-termination state.

Interpreting Inactivity Threshold Messages

Inactivity threshold messages help you track pipes, jobs that use pipes, and the BatchPipes subsystem. These messages appear on the operator's console when a reader or writer exceeds the threshold for one of the following states:

- Waiting-for-allocation
- Waiting-for-open
- Idle
- Waiting
- Waiting-for-close
- Waiting-for-EOF
- Waiting-for-termination.

For example, if the idle inactivity threshold setting is 20 minutes an job JOB8SUB, which writes to DATA1.TEMP2 using the subsystem BP01, is idle for 20 minutes, the following message appears:

```
ASFP407E BATCHPIPES BP01 WRITER JOB JOB8SUB IDLE ON PIPE DATA1.TEMP2
```

An inactivity threshold message appears only once per occurrence. In this example, no additional messages are displayed if JOB8SUB remains in the idle state for another 20 minutes.

When activity on the pipe begins again, BatchPipes issues the message:

```
ASFO408I BATCHPIPES BP01 WRITER JOB JOB8SUB NO LONGER IDLE ON PIPE DATA1.TEMP2
```

Similar messages appear for each inactivity threshold. The inactivity threshold messages (ASFP401 through ASFP422) are described in SmartBatch Messages and Codes.

For threshold messages, observe the following:

- If you see an exception message for most of your jobs, consider increasing the global value for the particular threshold. For example, if you specified 1 minute for the WAITOPEN threshold, this message will likely appear for most of your jobs.

  To modify threshold settings, do the following:

  – For parmlib threshold settings, use the BatchPipes DISPLAY command to check the settings. Use the BatchPipes SET command to modify these settings as required.

  – For pipe-level thresholds, you must change the value of the threshold on the SUBSYS parameter. The new value takes effect when the job is next submitted.

- If the same exception message is displayed for a particular job whenever the job is submitted, consider setting the appropriate pipe-level threshold for the pipe, rather than modifying the global setting for all jobs.
- If you have a job using a pipe in which you want to have a threshold message issued earlier than it would be if issued by the global threshold setting, use pipe-level thresholds. Suppose, for example, you have a job that should never experience a wait-for-open state. Here, you might have your global WAITOPEN setting at the default 15 minutes. You could then use the pipe-level threshold WAITOPEN to set the value to 0 for this specific pipe.
- It is possible that normally you do not expect to see threshold messages appear for a job. Day-to-day runs of your batch workload vary in the times that jobs start and finish. If this is not a common occurrence, you may choose to do nothing. Your staff should be familiar with the messages and understand their meanings.

An example of a threshold message might be for a reader that has been waiting for data beyond the WAIT threshold setting. This would prompt BatchPipes to issue this message:

```
  ASFP403E BATCHPIPES ssname READER JOB jobname WAITING FOR DATA ON PIPE pipename
```

Threshold messages might indicate that necessary jobs are not initiate or are waiting because of system activity. You should determine the cause the waits and take the appropriate actions. In this example, message ASFP403E tells the operator that a reader is in a wait threshold exception; that is, the reader has been waiting longer than the inactivity threshold value set for jobs in a waiting state. For this reader, one the following could have occurred:

- At least one writer job exists, but data is not flowing. Here, the writer is idle -- doing work unrelated to BatchPipes.

The appearance of message ASFP403E might cause an automated operations program to determine the jobname of the corresponding writer job and alert the operator to the delay. Or, it might trigger an operator command that improves the writer's performance specification.

Using the EOF Required Parameter

The EOFREQUIRED parameter allows you to specify what happens if a reader closes a pipe data set before it receives an EOF indication.

EOFREQUIRED=YES or NO

Specifies what happens if a reader closes a pipe data set before it receives an EOF indication. If you specify EOFREQUIRED=YES and a reader closes before it receives an EOF indication, the reader abends and all pipe users receive an I/O error.

If you specify EOFREQUIRED=NO, BatchPipes does not fail the reader if it closes the pipe data set before it receives an EOF indication. The default setting for EOFREQUIRED is NO.

The EOFREQUIRED subparameter on the JCL SUBSYS parameter overrides the EOFREQUIRED parameter specified in ASFPBPxx member. If you decide to use EOFREQUIRED and the number of jobs that close before the EOF indication is received is small, specify EOFREQUIRED=YES in ASFPBPxx member and specify EOFREQUIRED=NO on the JCL SUBSYS parameter for those jobs for which an EOF is not required.

Setting the Maximum BUFNO Value

On the MAXBUFNO parameter, you set a limit for the JCL BUFNO parameter that determines the pipe depth -- the number of buffers in the pipe.

MAXBUFNO=*nnn*

Specifies the maximum BUFNO value for pipes, where nnn ranges from 1 to 255. The default value for MAXBUFNO is 255.

Setting the Default BUFNO Value

On the DEFBUFNO parameter, you set the default value for the JCL BUFNO parameter that determines the PIPEDEPTH.

DEFBUFNO=*nnn*

Specifies the default BUFNO value for pipes, where nnn ranges from 1 to 255. The default value for DEFBUFNO is 7.

BatchPipeWorks DB2 Settings

On the DB2SSID parameter, set the default DB2 subsystem id to the BatchPipeWorks which will direct SQL requests. The default DB2 subsystem id is **DSN**.

On the DB2PLAN parameter, set the default DB2 plan that BatchPipeWorks will use to access DB2. The default DB2 plan is **BPWSQIP**.

Setting Up DB2 Access from BatchPipeWorks

Before you can use BatchPipeWorks to access DB2 data, steps must be taken so that the access is possible.

1. You must bind the DBRM (BPWSQIP shipped in SYS1.SASFPSAM) to a pla For information on how to bind the DBRM to the plan, see *DB2 Application Programming and SQL Guide*. The name of the plan that will bind is configurable and is described in this section. By default, the plan name used by BatchPipeWorks is BPWSQIP.

   ```
     Example of how to bind DBRM BPWSQIP to plan BPWSQIP from TSO
      READY
       DSN  SYSTEM(DSN)
      DSN
       BIND PLAN(BPWSQIP) MEMBER(BPWSQIP) ACTION(REPLACE) RETAIN
            VALIDATE(BIND) ISOLATION(CS) EXPLAIN(NO)

   (The partitioned data set that contains BPWSQIP must be allocated
   file DBRMLIB)
   ```

2. You must establish a default DB2 subsystem ID for BatchPipeWorks. If you plan to have the BatchPipes subsystem in operation, you may specify the default DB2 subsystem ID in the ASFPBPxx member of parmlib or use the BatchPipes SET command. The keyword to use is DB2SSID when setting the DB2 subsystem ID.

   If you do not plan to have the BatchPipes subsystem in operation, you can establish a default DB2 subsystem id by creating a persistent system-level name/token pair (refer to OS/390 MVS authorized assembler reference for instructions on how to create name/token pairs). The name portion of the name/token pair should be **BPWDB2SSID**. The default DB2 subsystem ID should be 1 to 4 characters, and should be the name of the primary DB2 subsystem that will be accessed with BatchPipeWorks. See *MVS/ESA Authorized Assembler Programming Guide* for information on creating name/token pairs.

   If you set no default, a default name of **DSN** will be used.

3. If you want to use a default plan name other than BPWSQIP, you must establish this plan name for BatchPipeWorks. If you plan to have the BatchPipes subsystem in operation, you can specify the default DB2 plan name in the ASFPBPxx member of parmlib or use the BatchPipes SET command. The keyword to use is DB2PLAN when setting the DB2 plan name.

   If you do not plan to have the BatchPipes subsystem in operation, you can establish a default DB2 plan name by creating a persistent system-level name/token pair. The name portion of the name/token pair should be **BPWDB2PLAN**. The default DB2 plan name should be 1 to 8 characters, and should be the name of the plan that the DBRM was bound to in step 1. See *MVS/ESA Authorized Assembler Programming Guide* for information on creating name/token pairs.

   If you set no default, a default name of **BPWSQIP** is used. IBM recommends that you use the default plan name.

4. You must grant access to the plan to those who will use it. For information on how this is done, see *DB2 for MVS/ESA Administration Guide*.

```
Example of granting authority to PUBLIC using BatchPipeWorks
 READY
PIPE LITERAL GRANT EXECUTE ON PLAN BPWSQIP TO PUBLIC]SQL
     NOCOMMIT EXECUTE ] TERMINAL
READY
```

Example of Initializing a BatchPipes Subsystem

A system programmer would take the following steps to get a BatchPipes subsystem named BP01 started:

1. Catalog 'SYS1.SASFPLIB' in the master catalog and catalog the system libraries.
2. Place the following procedure in SYS1.PROCLIB:

   ```
   //BP01 PROC
   //PSP EXEC PGM=ASFPMIKE,REGION=4096K,TIME=NOLIMIT,PARM='&PREFIX,&MEM'
   //STEPLIB DD DSN=SYS1.SASFPLIB,DISP=SHR
   //
   ```

   With BatchPipes installed, a sample procedure (member ASFPJI) exists in SYS1.SAMPLIB. You can copy and change that procedure.
3. Add SYS1.SASFPLIB, the name of the BatchPipes program library, to IEAAPFxx member of PARMLIB:

   ```
   .
   .
   .
   SYS1.SASFPLIB
   .
   .
   .
   ```
4. Activate SMF to record activity for three BatchPipes subsystems:

   ```
   .
   .
   .
   SUBSYS(BP01,TYPE(91))         /* Request All records for 'BP01'      */
   SUBSYS(BP02,TYPE(91(1:3)))  /* Request ONLY subsystem data for 'BP02'*/
   SUBSYS(BP03,TYPE(91(11:15)))  /* Request ONLY pipe data for 'BP03'   */
   .
   .
   .
   ```
5. Set SMF subsystem recording intervals for two BatchPipes subsystem and request no interval recording for a third subsystem:

   ```
   .
   .
   .
   SUBSYS(BP01,INTERVAL(003000))  /* Request 30-minute interval for 'BP01'*/
                                  /* Requires interval subtypes requested */
   SUBSYS(BP02,INTERVAL(006000))  /* Request 60-minute interval for 'BP02'*/
                                  /* Requires interval subtypes requested */
   SUBSYS(BP03,NOINTERVAL)    /* Request NO interval recording for 'BP03' */
   .
   .
   .
   ```

6. An installation planning to use the BatchPipes monitor would add the following DD statements to existing ISPF library concatenations in their TSO/E LOGON procs:

Figure 16. Example of Updating TSO/E LOGON Procedures

To this concatenation:	**Add the following DD statement:**
STEPLIB	DD DSN=SYS1.SASFPLIB,DISP=SHR...
SYSPROC	DD DSN=SYS1.SASFPEXE,DISP=SHR...
ISPPLIB	DD DSN=SYS1.SASFPPNL,DISP=SHR...
ISPMLIB	DD DSN=SYS1.SASFPTBL,DISP=SHR...
ISPTLIB	DD DSN=SYS1.SASFPTBL,DISP=SHR...
PIPEHELP	DD DSN=SYS1.SASFPHLP,DISP=SHR...

   This example assumes that BatchPipes has been installed in libraries having a high-level qualifier of "SYS1."

7. Use CSA=(a,b) in IEASYSxx member to reserve additional storage, if needed, for the BatchPipes subsystem, where a is CSA and b is ECSA Each BatchPipes subsystem uses approximately 3 kilobytes of CSA an 186 kilobytes of ECSA.

8. If you already have 55 system LXs defined in PARMLIB, increase the number of system LXs by the number your five BatchPipes subsystems will use. On the the NSYSLX keyword in IEASYSxx, specify the following:

```
.
.
.
NSYSLX=60
.
.
.
```

9. Add the names of the BatchPipes subsystems to IEFSSNxx member:

```
.
.
.
BP01                            NAME OF ONE BP SUBSYSTEM
BP02                            NAME OF ANOTHER SUBSYSTEM
.
.
.
```

   Planning ahead, the system programmer identifies a second BatchPipes subsystem 'BP02' that will be initialized later. This step allows a second BatchPipes subsystem to come up without a reIPL.

   You might have several BatchPipes subsystems active at a time for these uses:

   - To test future pipelines
   - To verify maintenance
   - To install and test a new release
   - To recreate and trace problems.

10. In the IEAICSxx member of PARMLIB, on the TRXNAME and PGN keywords specify the jobs that will be using BP01; for example:

    ```
    TRXNAME=BP1MOFE,PGN=119
    TRXNAME=BP1MODR,PGN=119
    ```

    The system programmer knew that PGN value 119 is available for use

11. In the IEAIPSxx member of PARMLIB, the following values might be set to define the performance group or groups assigned to the transactions in the ICS. The example is for a 5-job pipeline.

    ```
    /*          PERFORMANCE GROUPS                              */
    PGN=119,(DMN=12,DP=F6)
    ```

    Values vary according to changes in the use of processor, the I/O subsystem, and processor storage.

12. To allow a programmer with user ID BBROWN to run jobs that use BP01, enter the following RACF commands:

    ```
    RDEFINE FACILITY  PSP.ASFPPIPE.BP01 UACC(NONE)
    PERMIT PSP.ASFPPIPE.BP01 CLASS(FACILITY) ID(BBROWN) ACCESS(READ)
    ```

    The FACILITY class is already active so the SETROPTS does not have to be issued.

13. To allow a user named COMPTN to run authorized applications that have BatchPipeWorks fittings, enter the following RACF commands:

    ```
    RDEFINE FACILITY  PSP.ASFPAFIT.BP01 UACC(NONE)
    PERMIT PSP.ASFPAFIT.BP01 CLASS(FACILITY) ID(COMPTN) ACCESS(READ)
    ```

14. To use member ASFPBP00 to set inactivity thresholds, first copy member ASFPBP00 from SYS1.SAMPLIB to SYS1.PARMLIB. The following example sets the following inactivity threshold values:

    - An idle threshold of 5 minutes
    - Turn off the WAIT threshold so that no messages appear for wait jobs
    - A wait-for-open threshold of 5 minutes
    - A default BUFNO setting of 15.

    The settings in ASFPBP00 member of PARMLIB are:

    ```
    IDLE=5
    WAIT=OFF
    WAITOPEN=5
    DEFBUFNO=15
    ```

    Then, when you start the subsystem, you can specify that ASFPBP00 is the active member of PARMLIB. Or, you can create your own ASFPBP00 member set inactivity thresholds or use the BatchPipes SET command.

Example of Initializing a Second BatchPipes Subsystem

Initializing the succeeding instance of a BatchPipes subsystem requires that you consider all steps mentioned in the chapter. Because of the work you did in Step 8 when you initialized the first subsystem, the second subsystem is easier to initialize.

SMF Recording for BatchPipes

This chapter describes General-use Programming Interfaces and associated guidance information.

BatchPipes writes system management facilities (SMF) Type 91 (X'5B') records for BatchPipes. The records are written at START and STOP of the BatchPipes subsystem, at the subsystem recording interval set through SMFPRMxx parmlib member, at pipe creation and deletion, and at open and close of pipe connections. The records contain information about the BatchPipes subsystem and activity on the pipes. Specifically, the Type 91 record:

- Provides data that you can use to determine when BatchPipes activity approaches the number of pipe connections and allocations that BatchPipes allows per subsystem.
- Uniquely identifies the pipe. You can use the pipe id as a search argument to locate all the connections and associated activity for a pipe and to construct a pipeline topology. The record contains aggregate activity counts.
- Provides information that a scheduling product can use as a signal for scheduling a pipe partner, or for scheduling the rest of the pipeline.
- Describes the I/O activity of each connection on the pipeline.
- Collects data for a post-analysis routine to determine the effectiveness of the pipeline and make topology changes.
- Provides data from which you can calculate an effective chargeback unit based on resource consumption.

Type 91 records provide the following information:

- BatchPipes subsystem start and stop status, for example:
  - Number of pipe creations
  - Number of pipe deletions
  - Number of connections allocated
  - Number of opens (connects)
  - Number of closes (disconnects)
  - Status
  - Trace status.
- Pipe existence
  - First OPEN complete time and date
  - Last CLOSE time and date
  - Pipe DCB characteristics
  - Pipe I/O counts.
- OPEN and CLOSE connections
  - System name, subsystem name, jobname, stepname, ddname, pipename
  - Allocation date and time
  - Open date and time
  - OPEN DCB characteristics
  - Close date and time.

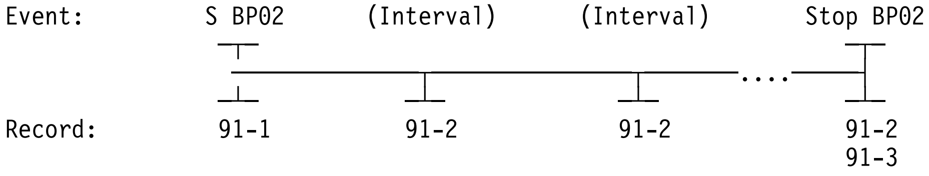
Subtypes of Type 91 Records

Type 91 records consist of eight subtypes:

- **Subtype 1, written at subsystem start**, contains data pertaining to the initialization of the subsystem.
- **Subtype 2, subsystem information written at subsystem recording interval expiration**, contains data that indicates changes in the numbers of pipes, allocations, connections, opens, and closes over the subsystem recording interval.
- **Subtype 3, written at subsystem stop**, contains data that indicates the status of the subsystem at subsystem ending. Depending on how the subsystem terminated, this record may not always be available. A subtype 2 record is written to indicate the activity between the last subtype 2 recording and subsystem ending.
- **Subtype 11, written at pipe open-connection**, contains open-connection date, time, DCB characteristics and user identification information
- **Subtype 12, pipe information written at subsystem recording interval expiration**, contains open-connection date, time, DCB characteristics, job identification information and a summary of activity for all the connections to the pipe.
- **Subtype 13, written at pipe close-connection**, contains open-connection date, time, DCB characteristics, close-connection date and time, user identification information and a summary of activity for connections to the pipe.
- **Subtype 14, written at pipe-creation**, is written when the first connection is made to a pipe. It contains general identification information about the pipe.
- **Subtype 15, written at pipe-deletion**, is written when the last connection to a pipe is closed; it contains general pipe information and a summary of activity for all connections to the pipe.

Example of Recording Subsystem Events

The following diagram shows the Type 91 recording for various events in the life of a BatchPipes subsystem named BP02.



An Example of Recording Events on Pipes

The following example shows SMF recordings of BatchPipes events where two users, writer U1 and reader U2, use a pipe named PIPE1. SMF records describe BatchPipes events, pipes, and users. (SMF record type and subtype are included in the example.)

In this example, U1 opens first and U2 opens shortly afterwards. When U1 finishes writing to the pipe it closes its connection. BatchPipes sends U2 an “end-of-file.” Later U2 empties the pipe and closes its connection.

```
         U1 Opens            U2 Opens           U1 Closes            U2 Closes
Event:    Pipe1  (Interval)   Pipe1  (Interval)   Pipe1     (Int)      Pipe1
           ─┬─                 ─┬─                 ─┬─                  ─┬─
            ├─────────┬─────────┼───────┬───────────┼─────────┬──────────┤
           ─┴─       ─┴─       ─┴─     ─┴─         ─┴─       ─┴─        ─┴─

Record:   91-14     91-12     91-11   91-12       91-12     91-12      91-12
          91-11      (U1)      (U2)    (U1,       (U1,U2)    (U2)       (U2)
           (U1)                         U2)       91-13                91-13
                                                   (U1)                 (U2)
                                                                       91-15
```

Other information relating to SMF recording of BatchPipes is:

- “Activating SMF Recording” on page 32page describes how to enable SMF.
- “Setting a Subsystem Recording Interval” on page 32page describes how to set the interval that allows SMF to give periodic reports.
- “Example of Initializing a BatchPipes Subsystem” on page 60pagecontains examples of enabling SMF for BatchPipes recording and setting subsystem recording interval.
- “ Appendix A: SMF Record Descriptions” on page 153page contains a complete description of all fields in Type 91 records.

Setting a Subsystem Recording Interval for BatchPipes

SMFPRMxx for BatchPipes

Record Type 91 subtypes 2 and 12 occur at an interval that you specify The interval is known as the subsystem recording interval. A subsystem recording interval applies only to BatchPipes.

Activate SMF interval recording in the SMFPRMxx member of PARMLIB in one of the following ways:

- Include the INTERVAL option specified on the SUBSYS statement. Include SUBSYS(bp01, ... INTERVAL(hhmmss) ..), where bp01 is the name of the BatchPipes subsystem and hhmmss is the recording interval.
- Omit SUBSYS(bp01, ... INTERVAL(hhmmss) ..) in SMFPRMxx to request a recording interval at the SYS level. Here, BatchPipes uses the values set on the SYS parameter that controls system-wide SMF recording.

Section III: Information for Operations Staff

Section III is organized into the following chapters:

- **"Chapter 7: Controlling the BatchPipes Subsystem"** page 69 describes the MVS and BatchPipes commands that control a BatchPipes subsystem.
- **"Chapter 8: Scheduling Considerations in a BatchPipes Environment"** page 79 describes how to reschedule the BatchPipes jobs and examine other jobs that might be affected by the concurrent running of the BatchPipes jobs.
- **"Chapter 9: Monitoring BatchPipes"** page 83 helps you use the commands that BatchPipes provides to monitor the state of jobs using BatchPipes, the pipes, and the BatchPipes subsystem.
- **"Chapter 10: Questions and Answers for the Operations Staff"** page 105 contains questions and answers on topics that interest the operations staff.

The information in this section is intended for those who might perform these tasks on behalf of BatchPipes:

- Control a BatchPipes subsystem
- Control job flow
- Monitor the status of pipes, a BatchPipes subsystem, and jobs using BatchPipes
- Respond to messages sent by BatchPipes
- Detect and respond to errors and slowdowns.

Chapter 7: Controlling the BatchPipes Subsystem

Control the BatchPipes subsystem through MVS and BatchPipes commands. Use the MVS START command to start the BatchPipes subsystem. Use one of the following two ways to stop a BatchPipes subsystem:

- The BatchPipes STOP command is a nondisruptive way of stopping a BatchPipes subsystem, allowing jobs to finish using the pipes associated with the subsystem.
- The BatchPipes CANCEL command stops a BatchPipes subsystem immediately and cancels all jobs that are using BatchPipes
- The BatchPipes SET command sets the inactivity threshold values for jobs using the BatchPipes subsystem.

Use the MVS FORCE BatchPipes-subsystem-name,ARM command as a last resort when the CANCEL command fails to perform its function, even after you have issued it several times.

CAUTION:
Never use the FORCE command without understanding that after issuing FORCE, you might have to reIPL.

All BatchPipes commands begin with the command prefix, *bp-cmd-prefix*. By default, the command prefix is the name of the BatchPipes subsystem. You can established a different prefix through the PARM parameter in the EXEC statement or by specifying PREFIX=*value* on the START command. See “Using the Name of the BatchPipes Subsystem” on page 30 page for more information on defining a command prefix.

To obtain a summary of all BatchPipes commands and their parameters issue the BatchPipes HELP command. The syntax is:

```
┌── BATCHPIPES HELP ──────────────────────────────────────────────────────┐
│                                                                          │
│ ►►─ bp-cmd-prefix───►──HELP──────────────────────────────────────────►◄ │
│                                                                          │
│                                                                          │
└──────────────────────────────────────────────────────────────────────────┘
```

Starting a BatchPipes Subsystem

Start the BatchPipes subsystem by issuing the MVS START command. For a BatchPipes subsystem, the syntax of the START command is as follows:

START *bp-subsystem-name*

Optional parameters you might use in any order on the START command are:

SUB=MSTR Requests that the BatchPipes subsystem address space not be started by JES2 or JES3. Use SUB=MSTR if you want the BatchPipes subsystem to continue running in the event JES2 or JES3 fails. If you do not code “bp-subsystem-name,ASFPMSSI” in IEFSSNxx member, then

SUB=MSTR is the default and does not need to appear on the START command.

PREFIX=pref Identifies the command prefix that you use to issue BatchPipes commands. By default, the command prefix for a BatchPipes subsystem's commands is the name of the subsystem. The prefix can be up to eight characters in length, with no embedded blanks. For example, if you name a subsystem BP01, you can establish a command prefix of @BP01 by coding the following parameter on the START command:

```
PREFIX=@BP01
```

MEM=xx Indicates which ASFPBPxx member of SYS1.PARMLIB is to be used by the BatchPipes subsystem that is starting. The ASFPBPxx member contains inactivity threshold values. The default member is ASFPBP00.

For the START command to succeed, the subsystem name must be in IEFFSSNxx member of SYS1.PARMLIB and a procedure with the same name must be in SYS1.PROCLIB.

Stopping a BatchPipes Subsystem

To stop a BatchPipes subsystem in a nondisruptive way, issue the BatchPipes STOP command.

```
┌── BATCHPIPES STOP ─────────────────────────────────────────────────────┐
│                                                                        │
│ ►►─ bp-cmd-prefix──►──STOP───────────────────────────────────────────►◄ │
│                                                                        │
└────────────────────────────────────────────────────────────────────────┘
```

You can also use the MVS STOP command and the MVS MODIFY command:

P *bp-subsystem-name* or F *bpsubsystem-name*,STOP

Jobs that are already using the pipes continue to run to completion. Jobs that try to connect to a pipe after the STOP command is issued fail and a messages such as the following appears:

```
IEF752I JOBA1 OUTDD - REQUEST FAILED BY SUBSYSTEM
ASFP318I BATCHPIPES ALLOCATION FAILURE. SPECIFIED SUBSYSTEM IS
     TERMINATING: JOBSTEP=JOBA1STP DD=OUTDD SUBSYS=BP01
```

Cancelling a BatchPipes Subsystem

To stop a BatchPipes subsystem immediately, use the BatchPipes CANCEL command. The syntax of the BatchPipes CANCEL command is:

```
┌─── BATCHPIPES CANCEL ──────────────────────────────────────────┐
│                                                                │
│ ►►─ bp-cmd-prefix──►──CANCEL─────────────────────────────────►◄ │
│                                                                │
│                                                                │
└────────────────────────────────────────────────────────────────┘
```

You can also use the MVS MODIFY command:

F *bp-subsystem-name*,CANCEL

All jobs using BatchPipes are cancelled, and the BatchPipes subsystem stops immediately.

- If the cancel action is not successful, the operator should reissue the BatchPipes CANCEL command. If the subsystem has stopped, but some jobs have not completed their cancel process, then the operator might have to explicitly cancel these outstanding jobs.

You can restart the subsystem immediately after the cancel action.

Note the difference between the STOP and the CANCEL commands. The STOP command allows the jobs to complete before the subsystem ends. The CANCEL command cancels the jobs associated with the subsystem before it stops the subsystem itself.

Sending an EOF Indication

To present an EOF indication to one or more readers on a specified pipe, use the BatchPipes EOF command. The syntax of the EOF command is as follows:

```
┌── BATCHPIPES EOF ──────────────────────────────────────────────────┐
│                                                                     │
►►─ bp01───EOF───,PIPE──=pname────────────────────────────────────►◄
│                                                                     │
└─────────────────────────────────────────────────────────────────────┘
```

Use the EOF command to present an EOF indication to one or more reader using the pipe named pname. The EOF command is effective only if all writers have closed the pipe data set and the last writer to close had the NOEOF subparameter specified on the JCL SUBSYS parameter. BatchPipes issues a message that tells the result of the command.

If the EOF command is entered while the pipe contains data, BatchPipes presents the EOF indication after all records in the pipe are read.

In a pipeplex, the BatchPipes EOF command indicates an end-of-file (EOF) condition to all readers of the specified pipe, regardless of the systems on which the readers reside.

Using the BatchPipes SET Command

Use the BatchPipes SET command to:

- Specify which ASFPBPxx member of PARMLIB is to be active.
- Override inactivity threshold values and other subsystem parameters set in ASFPBPxx. To see the ASFPBPxx parameters, see “Summary of Initialization Tasks” on page 49 page
- Set inactivity thresholds in ASFPBPxx that alert operations staff about inactivity of jobs. “Specifying an Alternate Parmlib Data Set for BatchPipes” on page 51 page describes the inactivity thresholds and offers some advice on how to set them. (The set parameter does not affect pipelevel thresholds that are in effect for a particular job.)
- Specify other subsystem parameters.

Figure 12-2 summarizes settings on the BatchPipes SET command. Capitalization of the parameters shows the shorthand version of the parameters. Following the table are the command syntax, definitions of parameters and examples.

Figure 17. Settings on the BatchPipes SET Command

To specify:	Use these parameters:
Which ASFPBPxx member of PARMLIB is active	MEMber=xx
Whether BatchPipes is to monitor the thresholds	THResh=ON or OFF

The inactivity threshold values	WAITALLOC=nnnn or OFF
	WAITOPEN=nnnn or OFF
	Idle=nnnn or OFF
	Wait=nnnn or OFF
	WaiTClose=nnnn or OFF
	WaitEOF=nnnn or OFF
	WaitTerm=nnnn or OFF

Whether readers abend if they close the pipe data set before receiving EOF notification	EOFRequired=YES or NO
The maximum BUFNO value	MAXBUFNO=nnn
The default BUFNO value	DEFBUFNO=nnn
The default DB2 subsystem name	DB2SSID=ssnm
The defaults DB2 plan name	DB2PLAN=planname

If the installation does not set a value in ASFBPBxx member of PARMLIB or on the SET command, BatchPipes uses the defaults mentioned in the parameter descriptions that follow.

Scope of the BatchPipes SET Command

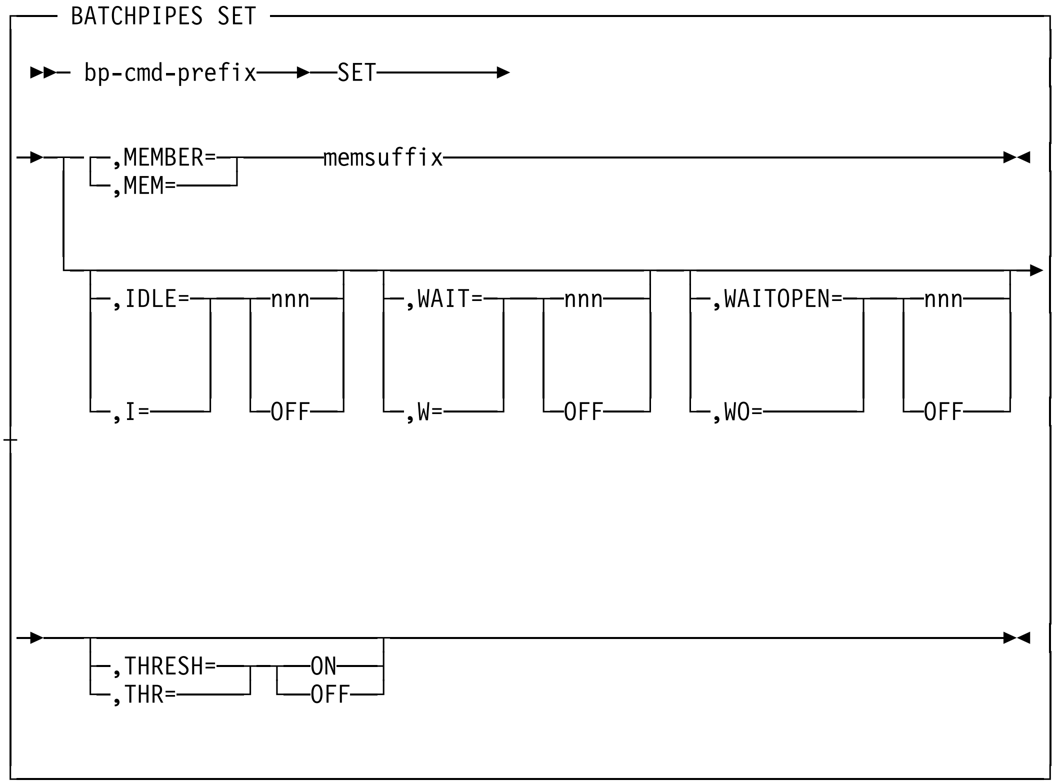
The SET command is pipeplex-wide in scope. That is, if you enter SET on one system in a pipeplex, the changes are propagated to all systems in the pipeplex.

Setting an Inactivity Threshold through the SET Command

Through the BatchPipes SET command, you can:

- Set inactivity thresholds
- Override the inactivity threshold values set in ASFPBPxx member
- Turn off the monitoring of inactivity thresholds, which means BatchPipes does not send inactivity threshold messages ASFP401 through ASFP412.

The syntax of the BatchPipes SET command



Parameters on the BatchPipes SET command are:

MEMBER=*xx*
MEM=*xx*
Specifies the ASFPBPxx member of SYS1.PARMLIB that the subsystem is to use.

IDLE=*nnn* or OFF
I=*nnn* or OFF
Specifies the inactivity threshold for idle jobs, where *nnn* is the number of minutes, ranging from 0 to 300, and OFF specifies that BatchPipes is not to detect inactivity threshold exceptions for idle jobs.

WAIT=*nnn* or OFF
W=*nnn* or OFF
Specifies the inactivity threshold for waiting jobs, where *nnn* is the number of minutes, ranging from 0 to 300, and OFF specifies that BatchPipes is not to detect inactivity threshold exceptions for waiting jobs.

WAITOPEN=*nnn* or OFF
WO=*nnn* or OFF
Specifies the inactivity threshold for jobs that are in a waiting-for-open state, where *nnn* is the number of minutes, ranging from 0 to 300, and OFF specifies that BatchPipes is not to detect inactivity threshold exceptions for jobs that are waiting-for-open.

THRESH=ON or OFF
THR=ON or OFF

Specifies whether BatchPipes is to monitor threshold exceptions. Use the THRESH parameter to temporarily suspend the monitoring of inactivity thresholds. THRESH=ON, the default, tells BatchPipes to monitor according to values specified on the IDLE, WAIT, or WAITOPEN parameters. THRESH=OFF turns off all monitoring for thresholds. If you subsequently set THRESH=ON, monitoring resumes using the values that existed at the time threshold monitoring was turned off.

An inactivity error message appears on the operator console for each inactivity threshold exception. Set threshold values low enough to provide helpful information, but high enough to avoid unnecessary messages. The default inactivity threshold is 15 minutes.

Examples of Setting Inactivity Threshold Values

Example 1: To set the idle threshold value to 10 minutes, the wait value to 5 minutes, and the wait-for-open value to 20 minutes, issue the following command:

```
BP01 SET,IDLE=10,WAIT=5,WAITOPEN=20
```

Example 2: To use the threshold values set in ASFPBP02 member of SYS1.PARMLIB, issue the following command:

```
BP01 SET,MEM=02
```

Checking the Values of the Inactivity Threshold

Use the BatchPipes DISPLAY command to learn the current settings of the inactivity thresholds:

bp-cmd-prefix DISPLAY

In response, message ASFP221I displays the inactivity threshold settings for jobs that are idle, waiting, and waiting-for-open. The message also indicates whether each threshold is a default value, a value set through ASFPBPxx member of SYS1.PARMLIB, or a value set through the BatchPipes SET command.

Authority to Issue BatchPipes Commands

The following BatchPipes commands require RACF CONTROL access:

- CANCEL
- DUMP
- SET
- STOP
- TRACE

The following BatchPipes commands require RACF READ access:

- DISPLAY
- HELP
- STATUS

```
BP120DR2  bp08 d
BP120DR2  ASFP221I 08:37:21 BP08 DISPLAY 787
    ACTIVE PARMLIB MEMBER:  ASFPBP00
    CURRENT THRESHOLD VALUES:
     IDLE=15   MINUTES   SOURCE:  PARMLIB
     WAIT=15   MINUTES   SOURCE:  PARMLIB
 WAITOPEN=15   MINUTES   SOURCE:  PARMLIB
WAITCLOSE=15   MINUTES   SOURCE:  DEFAULT
  WAITEOF=15   MINUTES   SOURCE:  DEFAULT
WAITALLOC=15   MINUTES   SOURCE:  DEFAULT
 WAITTERM=15   MINUTES   SOURCE:  DEFAULT
    CURRENT SUBSYSTEM SETTINGS:
    EOFREQUIRED=NO      SOURCE:  DEFAULT
       MAXBUFNO=255     SOURCE:  DEFAULT
       DEFBUFNO=7       SOURCE:  DEFAULT
       DB2SSID=DSN      DB2PLAN=BPWSQIP
```

Figure 18. Displaying the Inactivity Threshold Values and Subsystem Settings

Do not confuse the DISPLAY command with the STATUS command. The DISPLAY command has no parameters. Use it to learn the inactivity threshold values and other subsystem parameters.

Displaying the Pipeline Topology

With BatchPipes, several jobs and pipes can be interconnected. Using the FLOW parameter on the STATUS command, you can start at any specified job or pipe and display the status of all jobs or pipes interconnected to the specified job or pipe. The format of the command is:

```
bp01 STATUS,JOB=jname,FLOW
         or
bp01 STATUS,PIPE=pname,FLOW
```

The display is similar to that for a specified pipe. Under the line that identifies the name of the pipe, data flow information describes all writers using the pipe and then all readers using the pipe.

Displaying Flow of a Pipeline

A system programmer wants to know the jobs and pipes associated with JOBB. Figure 19 on page 77 shows pipes named PIPE.OUT1 and PIPE.OUT2 and the jobs using those pipes.

```
bp01 st,j=jobb,flow
ASFP210I 12:52:07 BP01 STATUS      056
PIPE=PIPE.OUT1                               DSPNM=ASFPDQ01
  JOB=JOBA1  SYS=SMF4 STEP=GENDATA  NUM=JOB00073
      WRITE       IDLE=00:00:00    COUNT=270      WAITS=0
         ACCUM I/O WAIT=00:00:00
  JOB=JOBA   SYS=SMF4 STEP=GENDATA  NUM=JOB00072
      WRITE       IDLE=00:00:00    COUNT=270      WAITS=0
         ACCUM I/O WAIT=00:00:00
  JOB=JOBB   SYS=SMF4 STEP=COPY     NUM=JOB00074
      READ        WAIT=00:00:00    COUNT=430      WAITS=0
         ACCUM I/O WAIT=00:00:00
  JOB=JOBC   SYS=SMF4 STEP=COPY     NUM=JOB00075
      READ        WAIT=00:00:34    COUNT=112      WAITS=3
         ACCUM I/O WAIT=00:00:38
PIPE=PIPE.OUT2                               DSPNM=ASFPDQ01
  JOB=JOBB   SYS=SMF4 STEP=COPY     NUM=JOB00074
      WRITE       IDLE=00:00:00    COUNT=423      WAITS=0
         ACCUM I/O WAIT=00:00:00
  JOB=JOBC   SYS=SMF4 STEP=COPY     NUM=JOB00075
      WRITE       IDLE=00:00:34    COUNT=105      WAITS=0
         ACCUM I/O WAIT=00:00:00
  JOB=JOBD   SYS=SMF4 STEP=REPORTS  NUM=JOB00076
      READ        WAIT=00:00:00    COUNT=529      WAITS=2
         ACCUM I/O WAIT=00:00:24
```

Figure 19. Displaying Flow of a Pipeline

The example assumes that a subsystem named BP01 is running and these pipes/jobs are open:

- Pipe PIPE.OUT1
  - Written to by JOBA1 and JOBA
  - Read from by JOBB and JOBC
- Pipe PIPE.OUT2
  - Written to by JOBB and JOBC
  - Read from by JOBD

This topology has seven connections, as numbered in Figure 20

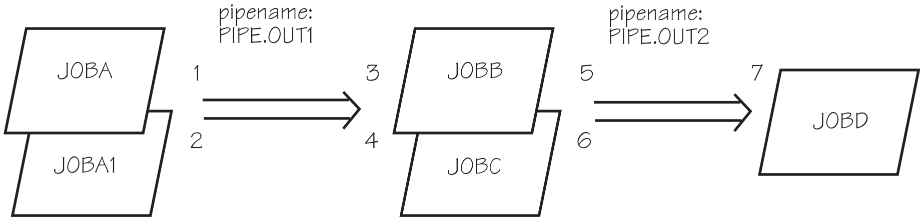


Figure 20. Pipeline with Seven Connections

When a pipe or job name is specified on the STATUS command, each connection results in two or three lines of status information.

Chapter 8: Scheduling Considerations in a BatchPipes Environment

Because all jobs within one pipeline must start at essentially the same time, changes to the current job scheduling process are required. The two timelines in Figure 21 contrast how JobX1, JobX2, and JobX3 run a non-BatchPipes and BatchPipes environment.

```
                 JOBS RUNNING WITHOUT BATCHPIPES

 <-------JobX1----------><-----JobX2------><--------JobX3--------->

mid-   2     4     6     8    10    noon   2     4     6     8    10
night

                 JOBS RUNNING WITH BATCHPIPES

 <--------JobX1---------->
  <----------JobX2--------->
   <-----------JobX3--------->

mid-   2     4     6     8    10    noon   2     4     6     8    10
night
```

Figure 21. Changing the Scheduling of Jobs for BatchPipes

You will need to reschedule jobs in the pipeline, as well as other jobs in the system — jobs that might contend for resources with the pipeline jobs and jobs that depend on successful completion of the pipeline jobs. Consider how to reschedule the non-BatchPipes work now that time is freed up by the rescheduled pipeline jobs.

```
                    NON-BATCHPIPES JOBS

                          <----This time is available for other---->
                            <--work, including jobs that depend---->
                              <-on completion of pipeline jobs.---->

mid-   2     4     6     8    10    noon   2     4     6     8    10
night
```

Figure 22. Changing the Scheduling of non-BatchPipes Jobs

Scheduling the BatchPipes Jobs

Take the following steps to evaluate and reschedule the BatchPipes jobs.

- Group the pipeline jobs into one scheduling group, or block; then, treat that block as a single entity.

  No job within the block should be submitted until **all** predecessor jobs for **all** the jobs in the block have completed. Once the predecessor jobs complete, submit all jobs in the block. This approach solves many problems associated with scheduling BatchPipes jobs.

  If a program requires all records before opening the output data set, perhaps you should have more than one scheduling group. For example, consider a sort job in the “middle” of a pipeline. Group all the jobs that run before the sort as a single entity. Then, group the jobs that follow the sort as a second entity. By scheduling sections of a pipeline in this manner, you can reduce concurrent resource allocations (that is, devices, initiators, auxiliary paging space, and datasets).

  In some cases it might be prudent to break the pipeline and direct the output to physical media. For example, if a sort is the “middle” stage of a many-stage pipeline, you could direct its output to tape or DASD. Although sending the data to physical media lowers the benefit of using pipes, it can serve as a recovery boundary and also serve to reduce the concurrent use of resources.

- Determine the best time to run the jobs using BatchPipes

  Estimate what the new elapsed time for the jobs will be. With the jobs running in less time, you have more flexibility in scheduling them.

  Estimate the impact of having additional processor storage, processor cycles, tape drives, DASD temporary storage (SORTWRK space), and required initiators to support the pipeline jobs. In certain situations, it might be best to delay running a BatchPipes application until off-peak hours. Then the jobs can be sure of having sufficient storage, tape drives, processor capacity to run in the least elapsed time. In other cases, it might be necessary to tolerate the performance impact of high resource contention to reach the ultimate goal of beating a deadline.

- Change input to your job scheduling program.

  For example, Operations Planning and Control/ESA (OPC/ESA) might be able to intercept threshold messages and issue alerts or take other actions to schedule jobs. These messages appear in response to waits, idles, or waiting-for-open situations that need some response. For information about thresholds, see “Setting Inactivity Thresholds through ASFPBPxx” on page 36.

- Select initiator class or classes for the jobs

  IBM recommends that you schedule all jobs using the same BatchPipes subsystem in the same initiator class. In a computing environment that includes more than one image of MVS, initiate the class only on the system that converts and executes the jobs using BatchPipes. In that way, you make sure the jobs run on the correct image of the system.

Scheduling the Non-BatchPipes Batch Jobs

Take the following steps to evaluate and reschedule the non-BatchPipes jobs in the system:

- Reschedule jobs that depend on successful completion of the pipeline jobs.

  Do not submit any of the jobs that depend on pipeline jobs until all jobs within the block complete successfully. Grouping the pipeline jobs together removes problems associated with rerunning the pipeline jobs, should that be necessary.

- Distribute the non-BatchPipes work

  With the BatchPipes jobs running concurrently in a smaller timeframe, other work that once ran along with those jobs have fewer resources available. You might need to reschedule that work to a timeframe that is freed up by BatchPipes running the jobs in a shorter time.

Considerations for Scheduling Multiple Writers or Readers

When a pipe has more than one writer or reader, make sure that the job and its clone or clones start at approximately the same time and that necessary resources are available. These resources include:

- Initiators
- Tapes and tape drives
- DASD
- Data sets

Consider one possible scenario of what happens if resources are not available. Let's look at the case where two writers and one reader are to use the same pipe. One writer and reader start. The second writer does not start; it waits for a tape mount. The first writer and reader finish processing; the writer closes the pipe and the reader receives an end-of-file indication. These actions result in the pipe connections closing. With the two active connections closed, BatchPipes deletes the pipe.

A late-arriving writer defines a new pipe with its open request and ends up waiting for a reader partner. Make sure there are no late-arriving jobs still waiting-to-open when the process is rescheduled.

Chapter 9: Monitoring BatchPipes

This chapter tells you three ways you can keep track of jobs using pipes, the pipes themselves, and the BatchPipes subsystem.

- The BatchPipes STATUS command allows you to take a “snapshot” of what is happening at a given moment. System programmers and performance analysts might use the command to learn how to improve the performance of the jobs using BatchPipes. The STATUS command is described in “Using the BatchPipes STATUS Command.”page
- Through the BatchPipes monitor that you use through ISPF. The monitor is described in “Using the BatchPipes Monitor” on page 96.
- Through threshold messages that appear on the operator's console when a specific job connected to a pipe is idle, waiting, or waiting-for-open. These messages are described in “Interpreting Inactivity Threshold Messages” on page 103 page and are listed in “ Appendix D: BatchPipes Messages” on page 203.page

Using the BatchPipes STATUS Command

Use the STATUS command to learn of activities related to your use of a BatchPipes subsystem: information about a pipe or pipes, a job or jobs, or the subsystem itself. Figure 17 on page 72 summarizes the options on the STATUS commands (all of which begin with *bp-cmd-prefix*) and tells you where you can find more information about the option.

Figure 23. Summary of the BatchPipes STATUS Command Options

To display information about:	See:
A specific job or jobs	page 85
A specific pipe or pipes	page 88
All pipes associated with a BatchPipes subsystem	page 91
A BatchPipes subsystem	page 92
Topology of the pipes and jobs	page 94

Data Flow Information

Whenever you request information about one or more jobs or pipes, you receive a line of information that describes the flow of data through the writer or reader connection. The line, called **data flow information**, looks like this:

```
                  WAIT=hh:mm:ss
                     or
WRITE or READ     IDLE=hh:mm:ss   COUNT=nnnnnnnn   WAITS=nnnnn
                     or
                  WAITOPEN=hh.mm.ss
```

Fields in the data flow information line follow:

WRITE or READ
Identifies whether the pipe connection is for a writer or a reader.

WAIT
A writer job has issued a write request to a full pipe or a reader job has issued a read request to an empty pipe. The time is the elapsed time since the request started, in hours, minutes, and seconds.

IDLE
A job is not issuing write or read requests, rather is performing an action that does not involve using BatchPipes. The time is the elapsed time since the last write or read request.

WAITOPEN
A job has issued an open to a pipe and is waiting-in-open for a partner to also open the pipe. The time is the elapsed time since open processing started, in hours, minutes, and seconds.

COUNT
The number of blocks that have entered the pipe or left the pipe since the job connected to the pipe. If the RECFM value is V or F, the number of blocks equals the number of records. This value may be as high as 999,999,999.

Notes on the COUNT value

You might find that the COUNT values for writer and reader are somewhat misleading. For example, it is possible for the read count to be greater than the write count by a very small number. Because BatchPipes formats writer data at one moment and reader data at another, this condition may occur. Figure 31 on page 95 shows two writers to PIPE.OUT2 have written a total of 528 blocks to the pipe, but the one reader has read a seemingly impossible 529 records (or blocks). The small difference should not undermine your confidence in the data.

WAITS
The count of how many times the job using this pipe connection has waited because of a “pipe full” condition (for a writer job) or a “pipe empty” condition (for a reader job).

Syntax of the STATUS Command

The syntax of the STATUS command is as follows:

```
┌── BATCHPIPES STATUS ──────────────────────────────────────────────────────┐
│                                                                            │
│ ►►──bp-cmd-prefix──►──┬─ST─────┬──────►                                    │
│                       └─STATUS─┘                                           │
│                                                                            │
│ ►──┬──┬──────────────────────────┬──────────┬──────────────────────────┬─┬──►◄
│    │  └─┬─,J=───┬─┬─jname───┬───┘          └─┬─,P=────┬─┬─pname───┬──┘ │  │
│    │    └─,JOB=─┘ ├─jname*──┤                └─,PIPE=─┘ ├─pname*──┤    │  │
│    │              └─*───────┘                           └─*───────┘    │  │
│    └──┬─┬─,J=───┬───jname────┬──,FLOW─────────────────────────────────┘  │
│       │ └─,JOB=─┘            │                                            │
│       └─┬─,P=────┬───pname───┘                                            │
│         └─,PIPE=─┘                                                        │
│                                                                            │
└────────────────────────────────────────────────────────────────────────────┘
```

You can also use the MVS MODIFY command to issue the BatchPipes STATUS command. For example, the command BP01 STATUS,J=EXJOB works the same as F BP01,STATUS,J=EXJOB, where BP01 is the name of the subsystem.

Displaying Information about Specific Jobs

To display information about specific jobs that are using BatchPipes, issue the following:

bp-cmd-prefix STatus,Job=*jname*|*jname**|*

where the *jname** specifies all jobs with names that begin with *jname* and "*" specifies all jobs running under that BatchPipes subsystem.

The information that appears in response is arranged by job:

```
ASPF210I . . .
J=jname   S=stepname   N=jobid
 P=pname                               DSPNM=dataspace
    WRITE       IDLE=hh:mm:ss    COUNT=nnnnnnnn  WAITS=nnnnn
 P=pname                               DSPNM=dataspace
    READ        WAIT=hh:mm:ss    COUNT=nnnnnnnn  WAITS=nnnnn
 P=pname                               DSPNM=dataspace
    READ     WAITOPEN=hh:mm:ss    COUNT=nnnnnnnn  WAITS=nnnnn
```

For each specified job, the following information appears after the heading that contains the message number: the job name, step name, and job id. The job id is a numerical identifier with a 1- to 3-letter prefix that indicates one of the following types of jobs:

- Batch job -- job id is of the form "JOBnnnnn."
- Started task -- job id is of the form "STCnnnnn"
- An APPC transaction -- job id is of the form "Annnnn"

Two lines appear for each pipe with connections to the job:

- The first line tells:
  - The pipe name.
  - The data space name used for the pipe. (The data space name is of interest only to IBM service personnel.)
- The second line contains data flow information for the pipe connection. See "Data Flow Information" on page 83.

If you request either the status of a job not connected to any pipe or the status of a pipe not connected to any job, the system returns the following message:

```
ASFP210I 14.09.02 BP01 STATUS
ASFP209I SPECIFIED PIPE/JOB UNKNOWN TO BP01
```

Example 1: Displaying Status by Job: An operator receives a call from a user who wants to know why the step named REPORTS in the job named PGM147 has not finished running. The operator suspects that the job is using pipes.

Figure 24 shows the command the operator issues and the display that appears.

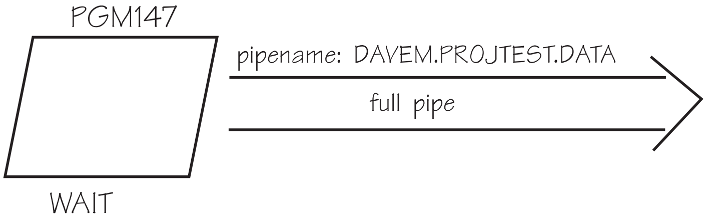
```
bp01 st,j=pgm147
ASFP210I 14:10:03 BP01 STATUS      181
J=PGM147   S=REPORTS   N=JOB00081
  P=DDAVM.PROJTEST.DATA                          DSPNM=ASFPDQ01
      WRITE        WAIT=00:05:25    COUNT=38209  WAITS=5
```

Figure 24. Example 1: Displaying Status of a Specific Job

This display tells you the following information about PGM147's use of the pipe:

- The job PGM147 is a writer to the pipe named DDAVM.PROJTEST.DATA.
- 38,209 records have already been written to the pipe since the pipe was opened by this job.
- The job has been waiting 5 minutes and 25 seconds since the last write operation was requested.
- During the course of processing, the job has had to wait on 5 different occasions due to a "pipe full" condition.

You might understand the display more clearly through the following illustration:



```
     Data flow information for writer:
 - WAIT for 5 minutes, 25 seconds with the pipe full
 - 38,209 records have been written
 - The job has waited 5 times due to the pipe being full
```

You need to ask some questions about the reader. The display gives us lots of proof that a reader exists: the count field shows records have passed through the pipe and the WAITS field shows indication of activity. Has the reader run to completion? Is it looping, or waiting for another event? To get answers to these questions, issue BP01 STATUS,PIPE=DDAVM.PROJTEST.DATA to find out the name of the reader from that pipe. Or, issue BP01 STATUS,JOB=PGM147,FLOW to see the topology of the pipeline.

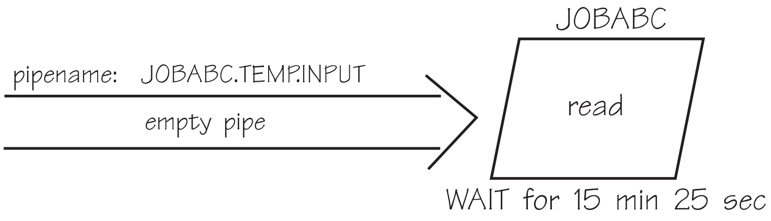
Example 2: Displaying Status by Job: Another user wants to know what is happening to a job named JOBABC.

Figure 25 shows the command the operator issues and the display that appears.

```
bp01 st,j=jobabc
ASFP210I 14:10:03 BP01 STATUS      181
J=JOBABC    S=STEP2     N=JOB32064
  P=JOBABC.TEMP.INPUT                            DSPNM=ASFPDQ01
      READ         WAIT=00:15:25    COUNT=0     WAITS=1
```

Figure 25. Example 2: Displaying Status of a Specific Job

This display tells you that over 15 minutes ago, a reader named JOBABC requested data from the pipe and has been waiting since then. Because no data has transferred since the pipe was open, the operator can conclude that the writer job is now idle — perhaps waiting for a tape mount, or a WTOR, or for TCP/IP data to arrive.



```
     Data flow information for reader:
 - WAIT for 15 minutes, 25 seconds with the pipe empty
 - No records have been read
 - The job has waited 1 time due to an empty pipe
```

Let's guess at what is happening to cause the scenario where only the reader is requesting the use of the pipe. Perhaps the operator needs to cancel the reader; or perhaps the writer is waiting for an operator reply or some other action.

To confirm the guesses, you could display the status of the pipe named JOBABC.TEMP.INPUT and learn about the writer. See "Example 3: Displaying

Status by Pipe" on page 88 for the results of that command. Or, you could issue BP01 STATUS,P=JOBABC.TEMP.INPUT,FLOW to see the topology of the pipeline.

Displaying Information about a Specific Pipe

You can request information about one pipe or all pipes. You can get a subset of pipes by using an asterisk ("*"). The following command displays information about the pipe or pipes.

bp-cmd-prefix STatus,Pipe=*pname*|*pname**|*

where *pname** specifies all pipes with names that begin with *pname* and "*" specifies all pipes associated with that BatchPipes subsystem.

The display that responds to this STATUS command contains information about the pipe or pipes. It is similar to that described in "Displaying Information about Specific Jobs" on page 85, except information is grouped by pipe name.

```
ASPF210I . . .
P=pname                                    DSPNM=ASFPDQnn
  J=jname    S=sname    N=jobid
     WRITE       IDLE=hh:mm:ss    COUNT=nnnnnnnn    WAITS=nnnnn
  J=jname    S=sname    N=jobid
     READ        WAIT=hh:mm:ss    COUNT=nnnnnnnn    WAITS=nnnnn
```

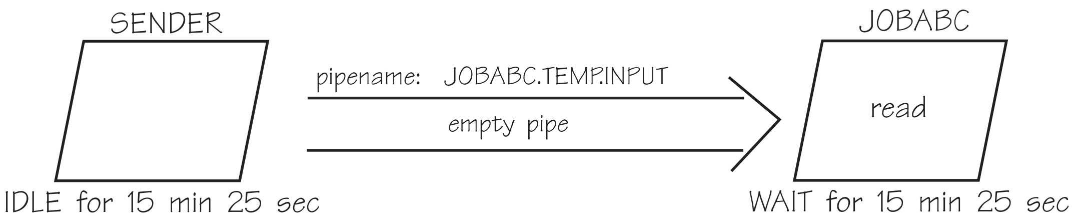
This description of a pipe shows one writer and one reader. (Note that a pipe might have more than one writer and more than one reader.)

Example 3: Displaying Status by Pipe: This example continues the actions of "Example 2: Displaying Status by Job" on page 87. The operator wants information about the pipe named JOBABC.TEMP.INPUT — the name that appeared in the display in Example 2. Figure 26 shows the command the operator issues and the display that appears.

```
bp01 st,p=jobabc.temp.input
ASFP210I 14:21:26 BP01 STATUS      251
P=JOBABC.TEMP.INPUT                                  DSPNM=ASFPDQ01
  J=SENDER   S=         N=JOB00001
      WRITE        IDLE=00:15:25    COUNT=0        WAITS=0
  J=JOBABC   S=STEP3    N=JOB00005
      READ         WAIT=00:15:25    COUNT=0        WAITS=1
```

Figure 26. Example 3: Displaying Status of a Specific Pipe

This display tells you about the pipe named JOBABC.TEMP.INPUT, as shown in the following illustration:



Two jobs are using the pipe; one is a writer named SENDER and the other is a reader named JOBABC. The writer has been idle for 15 minutes 25 seconds. Meanwhile, the reader has been attempting to read a record from the pipe for over 15 minutes 25 seconds, but the pipe is empty. As indicated by WAITS=1, this is the first time the reader has waited due to an empty pipe.

Both jobs are running and the writer hasn't added to the pipe for at least 15 minutes. The problem might be one of the following:

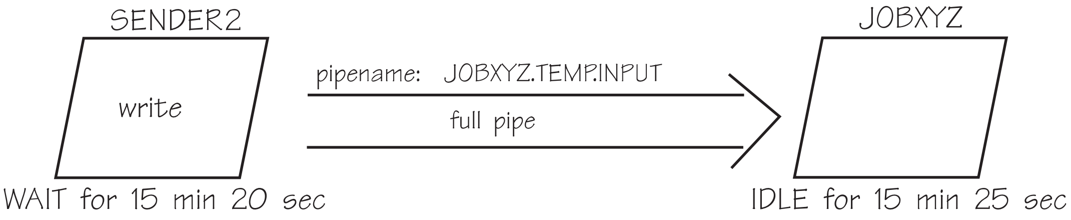
- A serialized resource is not available
- An application error in SENDER's code
- SENDER is doing other work
- SENDER has been pre-empted by a higher priority job
- SENDER has been swapped out
- SENDER is waiting on a tape mount for a non-piped dataset.

Example 4: Displaying Status of a Specific Pipe: Another possible result from the BP01 STATUS,PIPE command is in Figure 27, where the name of the pipe is JOBXYZ.TEMP.INPUT.

```
bp01 st,p=jobxyz.temp.input
ASFP210I 14:21:26 BP01 STATUS      251
P=JOBXYZ.TEMP.INPUT                                  DSPNM=ASFPDQ01
  J=JOBXYZ   S=STEP2    N=JOB00009
      READ         IDLE=00:15:25    COUNT=0       WAITS=0
  J=SENDER2  S=         N=JOB00006
      WRITE        WAIT=00:15:20    COUNT=7       WAITS=1
```

Figure 27. Example 4: Displaying Status of a Specific Pipe

This display tells you about the pipe named JOBXYZ.TEMP.INPUT, as shown in the following illustration:



The entry for the individual jobs tells you that SENDER2 (the writer) has successfully written seven records and has unsuccessfully tried to write the eighth, finding a full pipe. That last write request was issued 15 minutes 20 seconds ago, but the reader has not read from the pipe for 15 minutes 25 seconds. As indicated by WAITS=1, this is the first time the writer has waited due to a full pipe.

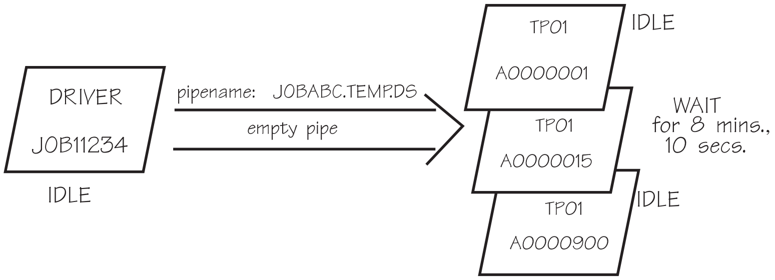
Determine whether JOBXYZ is waiting for some non-BatchPipes event.

Example 5: Displaying Status by Pipe: Check the status and performance of all jobs connected to the pipe named JOBABC.TEMP.DS.

```
bp01 st,p=jobabc.temp.ds
ASFP210I 14:21:26 BP01 STATUS      251
P=JOBABC.TEMP.DS                                      DSPNM=ASFPDQ01
  J=TP01     S=        N=A0000001
      READ         IDLE=00:00:01    COUNT=7778      WAITS=0
  J=TP01     S=        N=A0000015
      READ         WAIT=00:08:10    COUNT=310       WAITS=100
  J=TP01     S=        N=A0000900
      READ         IDLE=00:00:00    COUNT=5147      WAITS=0
  J=DRIVER   S=        N=JOB11234
      WRITE        IDLE=00:00:00    COUNT=13235     WAITS=0
```

Figure 28. Example 5: Displaying Status by Pipe

The display tells you that the pipe named JOBABC.TEMP.DS has three readers and one writer; one reader is waiting for a record from the pipe.



The writer, a batch job, has three jobs reading from the pipe. The three readers are APPC jobs with job ids A0000001, A0000015, and A0000900.

- A0000015 has been waiting for 8 minutes, 10 seconds.
- A0000001 and A0000900 are idle.

The 8-minute wait might need investigating. Let's look at the data flow information for reader A0000015 that has been waiting for records:

```
WRITE        WAIT=00:08:10    COUNT=310        WAITS=100
```

This reader has only read 310 blocks as compared with the other two readers that have read over 12,000 records. Also, this reader has waited 100 times due to the pipe being empty whenever it attempted to read. Based on this, the reader is unnecessary. Instead of running three reader jobs, try running only two reader jobs next time. For more information about how multiple readers read from a pipe, see “Considerations With Multiple Writers or Readers” on page 112.

Displaying Information about All Pipes

You can use the BatchPipes STATUS command to display information about all pipes associated with a particular subsystem. Use the following command:

bp-cmd-prefix STatus,Pipe=*

where *bp-cmd-prefix* is the target subsystem.

The display lists all pipes associated with the subsystem and all jobs using those pipes.

Example 6: Displaying Status of All Pipes: The following example shows the status of all pipes associated with the BatchPipes subsystem named BP01.

```
bp01 status,p=*
ASFP210I 11:24:17 BP01 STATUS       430

 P=DATA1.OUT01                                         DSPNM=ASFPDQ01
   J=ACCTJ4A  S=IBM      N=JOB00079
       READ     WAITOPEN=00:01:33    COUNT=0        WAITS=0
 P=MDAT.OUT02                                          DSPNM=ASFPDQ01
   J=ACCT01   S=RINGER   N=JOB00076
       WRITE        IDLE=00:00:00    COUNT=2693     WAITS=0
   J=ACCTJ2A  S=IBM      N=JOB00078
       READ         WAIT=00:00:00    COUNT=2694     WAITS=2
 P=MDAT.OUT01                                          DSPNM=ASFPDQ01
   J=ACCT01   S=RINGER   N=JOB00076
       WRITE        IDLE=00:00:00    COUNT=2700     WAITS=0
   J=ACCTJ1A  S=IBM      N=JOB00077
       READ         WAIT=00:00:00    COUNT=2701     WAITS=5
 P=P1                                                  DSPNM=ASFPDQ01
   J=JOPIPW1  S=WRITER   N=JOB00074
       WRITE        IDLE=00:00:00    COUNT=12809    WAITS=0
   J=JOPIPR1  S=READER   N=JOB00075
       READ         IDLE=00:00:00    COUNT=12808    WAITS=0
```

Figure 29. Example 6: Displaying Status of All Pipes

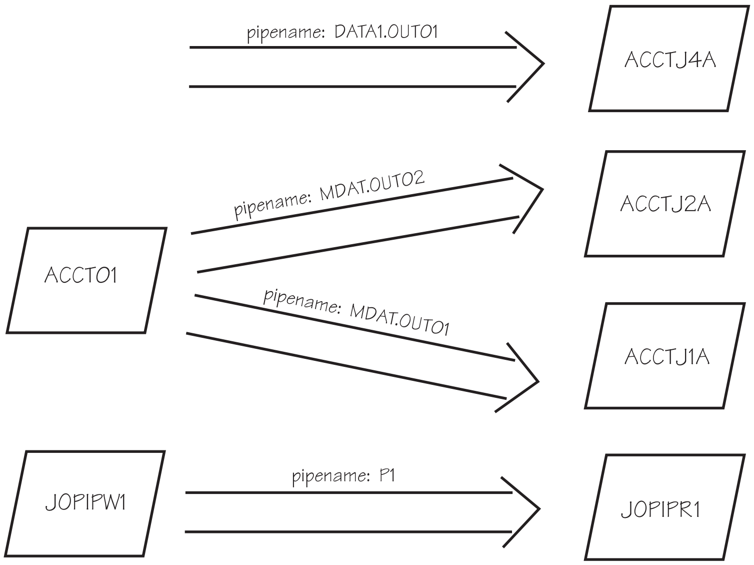
This display tells you that six jobs (ACCTJ4A, ACCT01, ACCTJ2A, ACCTJ1A, JOPIPW1, and JOPIPR1) are using four pipes:

- DATA1.OUT01
  - One reader ACCTJ4A
  - No records have moved to the pipe
  - The reader is waiting-in-open for a partner to open the pipe.
- MDAT.OUT02
  - One writer ACCT01
  - One reader ACCTJ2A
  - Over 2600 records have moved through the pipe
  - Reader ACCTJ2A has waited 2 times due to an empty pipe.
- MDAT.OUT01
  - One writer ACCT01
  - One reader ACCTJ1A
  - About 2700 records have moved through the pipe

  - Reader ACCTJ1A has waited 5 times due to an empty pipe.
- P1
  - One writer JOPIPW1
  - One reader JOPIPR1
  - Over 12,000 records have moved from the pipe.

Note that the job ACCT01 writes to two pipes.

The following illustration describes the pipes associated with the subsystem BP01:



Displaying Information about a BatchPipes Subsystem

If you want to know about the general status of a BatchPipes subsystem, use the following command:

bp-cmd-prefix STatus

The display that appears in response has a summary entry for the subsystem.

```
ASFP210I . . .
     #JOBS=nnnnn  #OPEN=nnnnn   #ALLOC=nnnnn       STATUS=status
 #WRITERS=
          #IDLE=nnnnn        MAX IDLE TIME=hh:mm:ss
          #WAIT=nnnnn         MAX WAIT TIME=hh:mm:ss
     #WAITOPEN=nnnnn      MAX WAITOPEN TIME=hh:mm:ss
 #READERS=
          #IDLE=nnnnn        MAX IDLE TIME=hh:mm:ss
          #WAIT=nnnnn         MAX WAIT TIME=hh:mm:ss
     #WAITOPEN=nnnnn      MAX WAITOPEN TIME=hh:mm:ss
```

The first line after the message identifier shows:

- #JOBS - The number of jobs that have allocated at least one pipe that is controlled by the subsystem
- #OPEN - The number of open pipes controlled by the subsystem. Note that even though multiple jobs may have opened a given pipe, in this number, the pipe is counted as open only once.
- #ALLOC - The number of data set allocations for pipes controlled by the subsystem (as stated on the DD statements). Note that the number of allocations always equals or exceeds the number of opens.
- STATUS - Status of the subsystem. The two options are:
  - ACTIVE
  - STOPPING - Displayed only when an operator has issued the BatchPipes STOP or CANCEL command, but the subsystem has not yet stopped.

The second through fifth lines give information about writers:

- #WRITERS - The number of writers to one or more pipes. Each instance of a job connected to a pipe for output is considered to be a writer.
- #IDLE - The number of writers that are idle. A writer is said to be idle if it is not writing to the pipe (that is, has no outstanding write requests).
- MAX IDLE TIME - The maximum time a writer has ever been idle. A large idle time most likely indicates the job is waiting for some non-BatchPipes event.
- #WAIT - The number of writers waiting for an outstanding write request to complete because the pipe is full.
- MAX WAIT TIME - The maximum amount of time any one writer has ever waited when attempting to write to a pipe that is full. A large time probably indicates that a reader partner has already ended, or is simply not reading from the pipe, such as waiting for a tape mount.
- #WAITOPEN - The number of writers that are waiting-in-open for a reader partner.
- MAX WAITOPEN TIME - The maximum amount of time a writer has ever had to wait-for-open for a reader partner. A large time probably indicates that a reader partner has not been started, has already ended, or has not yet opened the pipe for input.

The sixth through ninth line gives information about readers:

- #READERS - The number of readers of one or more pipes. Each instance of a job connected to a pipe for input is considered to be a reader.
- #IDLE - The number of readers that are idle. A reader is said to be idle if it is not reading from the pipe (that is, has no outstanding read requests).
- MAX IDLE TIME - The maximum time a reader has ever been idle. A large idle time usually indicates that the job is waiting for some non-BatchPipes event.
- #WAIT - The number of readers waiting for an outstanding read request to complete because the pipe is empty.
- MAX WAIT TIME - The maximum amount of time any reader has ever been waiting while attempting to read from an empty pipe. A large time probably indicates that a writer partner has already ended, or is not writing to the pipe, for example, waiting for a tape mount.
- #WAITOPEN - The number of readers that are waiting-in-open for a writer partner.
- MAX WAITOPEN TIME - The maximum amount of time a reader has ever had to wait-for-open for a writer partner. A large time probably indicates that a

writer partner has not been started, has already ended, or has not yet opened the pipe for output.

Example 7: Display Subsystem Information: The operator wants to restart the subsystem and wants to know how many jobs will be affected by a BatchPipes CANCEL of the subsystem named BP01.

Figure 30 shows the command the operator issues and the display that appears.

```
bp01 status
ASFP210I 11:23:59 BP01 STATUS       424
     #JOBS=6         #OPEN=3       #ALLOC=7           STATUS=ACTIVE
  #WRITERS=3
           #IDLE=3            MAX IDLE TIME=00:00:13
           #WAIT=0            MAX WAIT TIME=00:00:15
       #WAITOPEN=0        MAX WAITOPEN TIME=00:00:45
  #READERS=4
           #IDLE=1            MAX IDLE TIME=00:00:15
           #WAIT=2            MAX WAIT TIME=00:00:10
       #WAITOPEN=1        MAX WAITOPEN TIME=00:01:33
```

Figure 30. Example 7: Displaying Status of a BatchPipes Subsystem

The example shows that six jobs will be affected if the operator cancels the subsystem BP01. Among the six jobs, three are writer jobs and four are reader jobs; one of the writers is writing to two pipes. Three pipes are open. All three writers are idle (that is, no write requests are outstanding). One reader is idle (that is, no read requests are outstanding), one is waiting-in-open for a writer partner, and two are waiting on empty pipes.

The topology of the pipes in this example is described in "Displaying Information about All Pipes" on page 91, where one of the jobs writes to two pipes.

Displaying the Pipeline Topology

With BatchPipes, several jobs and pipes can be interconnected. Using the FLOW parameter on the STATUS command, you can start at any specified job or pipe and display the status of all jobs or pipes interconnected to the specified job or pipe. The format of the command is:

bp-cmd-prefix STatus,J=*jname*,FLOW
        or
bp-cmd-prefix STatus,P=*pname*,FLOW

The display is similar to that for a specified pipe. Under the line that identifies the name of the pipe, data flow information describes all writers using the pipe and then all readers using the pipe.

Example 8: Displaying Flow of a Pipeline: A system programmer wants to know the jobs and pipes associated with JOBB. Figure 31 on page 95 shows pipes named PIPE.OUT1 and PIPE.OUT2 and the jobs using those pipes.

```
bp01 st,j=jobb,flow
ASFP210I 12:52:07 BP01 STATUS      056
P=PIPE.OUT1                                            DSPNM=ASFPDQ01
  J=JOBA1    S=GENDATA  N=JOB00073
      WRITE        IDLE=00:00:00    COUNT=270      WAITS=0
  J=JOBA     S=GENDATA  N=JOB00072
      WRITE        IDLE=00:00:00    COUNT=270      WAITS=0
  J=JOBB     S=COPY     N=JOB00074
      READ         WAIT=00:00:00    COUNT=430      WAITS=0
  J=JOBC     S=COPY     N=JOB00075
      READ         WAIT=00:00:34    COUNT=112      WAITS=3
P=PIPE.OUT2                                            DSPNM=ASFPDQ01
  J=JOBB     S=COPY     N=JOB00074
      WRITE        IDLE=00:00:00    COUNT=423      WAITS=0
  J=JOBC     S=COPY     N=JOB00075
      WRITE        IDLE=00:00:34    COUNT=105      WAITS=0
  J=JOBD     S=REPORTS  N=JOB00076
      READ         WAIT=00:00:00    COUNT=529      WAITS=0
```

Figure 31. Example 8: Displaying Flow of a Pipeline

The example assumes a subsystem named BP01 is running and these pipes/jobs are open:

- Pipe PIPE.OUT1
  - Written to by JOBA1 and JOBA
  - Read from by JOBB and JOBC
- Pipe PIPE.OUT2
  - Written to by JOBB and JOBC
  - Read from by JOBD

This topology has seven connections, as numbered in Figure 32.

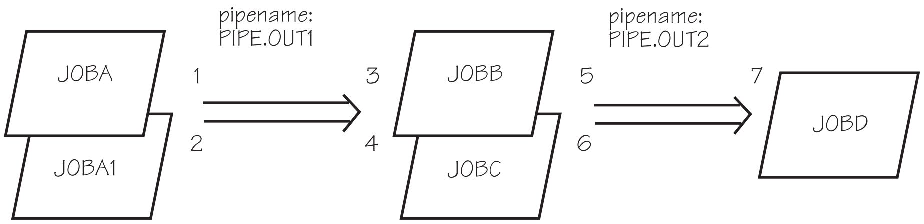


Figure 32. Pipeline with Seven Connections

When a pipe or job name is specified on the STATUS command, each connection results in two or three lines of status information.

Using the BatchPipes Monitor

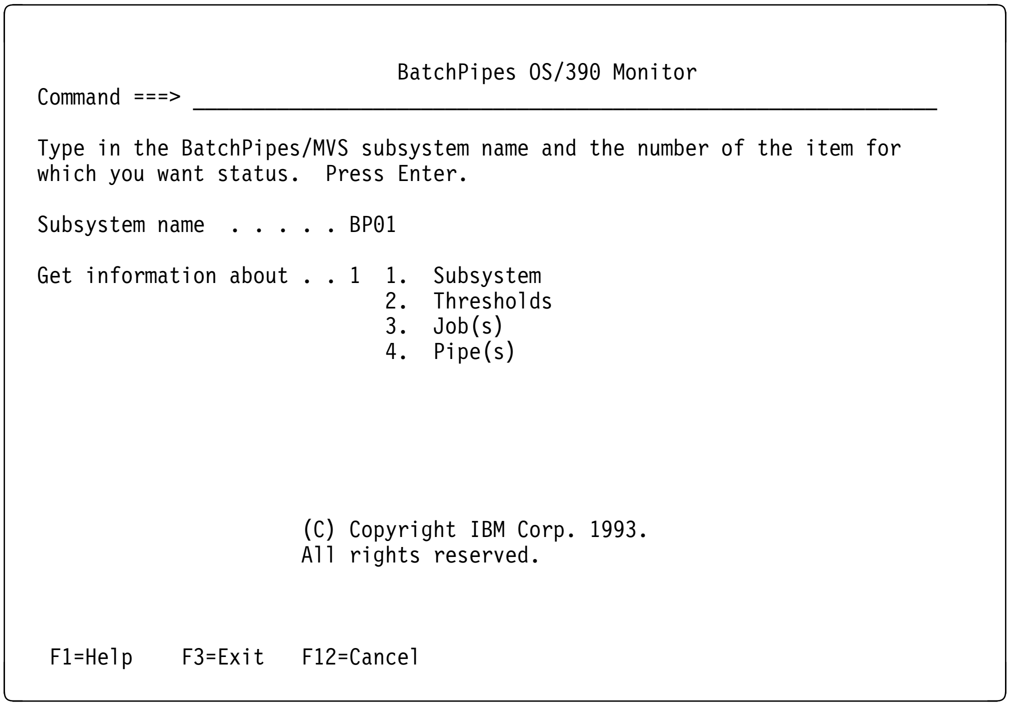
The BatchPipes monitor consists of a series of panels that provide information about jobs, pipes and a BatchPipes subsystem. Through ISPF panels, a TSO user can view the information. In addition, a comprehensive help system is available as part of the panel system. The following sections describe the panels and how to use them effectively.

ISPF Version 3 Release 3 or higher is required to use the BatchPipes monitor. Be sure the BatchPipes monitor is installed; for details about installing the monitor, see the program directory that accompanies the product.

To use the monitor, your LOGON PROC must be changed. You can restrict use of the monitor through RACF by protecting the BatchPipes libraries that are included in the LOGON PROC concatenations. The program directory that accompanies BatchPipes describes the changes to the LOGON PROC and includes details of the BatchPipes libraries.

Invoking the BatchPipes Monitor

Invoke the BatchPipes monitor through the TSO commands option of ISPF by entering ASFPMEMN on the command line. This command displays the main panel of the monitor as in Figure 33.



```
                              BatchPipes OS/390 Monitor
Command ===> ______________________________________________________________

Type in the BatchPipes/MVS subsystem name and the number of the item for
which you want status.  Press Enter.

Subsystem name  . . . . . BP01

Get information about . . 1  1.  Subsystem
                             2.  Thresholds
                             3.  Job(s)
                             4.  Pipe(s)





                      (C) Copyright IBM Corp. 1993.
                      All rights reserved.



 F1=Help    F3=Exit   F12=Cancel
```

Figure 33. BatchPipes Monitor Main Panel

On the main panel, fill in the name of a BatchPipes subsystem and the function desired. **Pressing PF1 on any panel displays the associated HELP panel.**

The four functions of the BatchPipes monitor and the pages on which you can find descriptions of the panels are:

- Subsystem information, described on page 97

- Inactivity threshold information, described on page 98

  An inactivity threshold exception is a job that has been in a state of waiting, idle, or waiting-to-open for longer than a specified period of time, known as the inactivity threshold. For information about how to set the inactivity threshold, see “Setting Inactivity Thresholds through ASFPBPxx” on page 36.

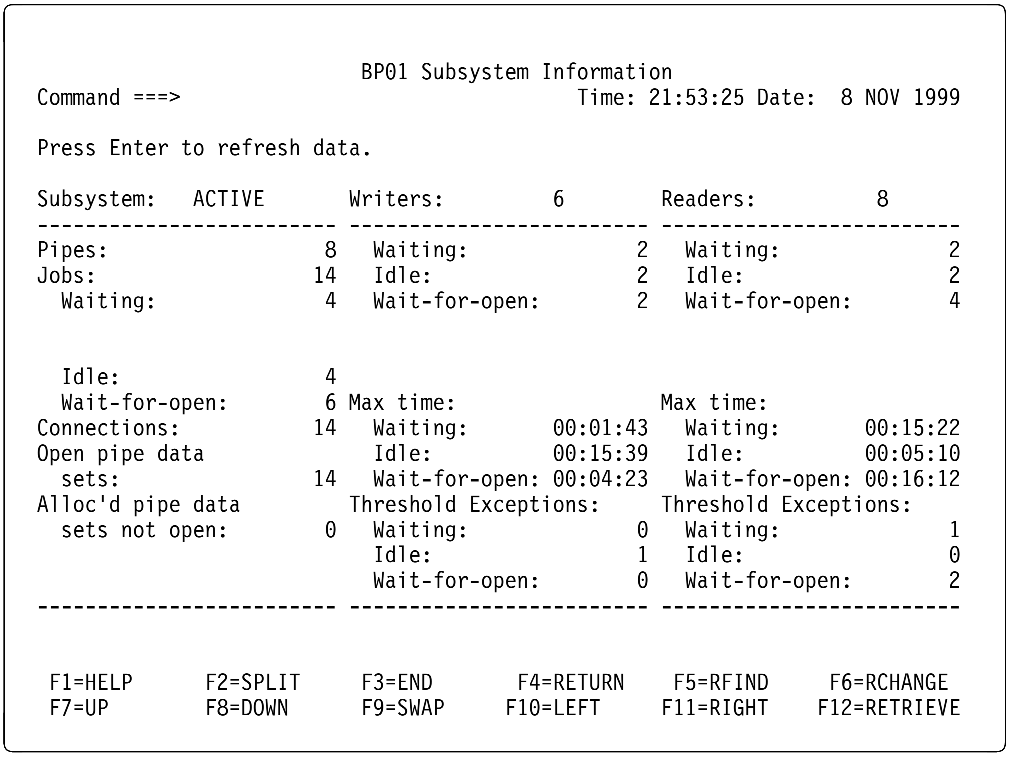
- Job information described on page 100
- Pipe information, described on page 102.

Subsystem Information

To obtain summary information about a BatchPipes subsystem, select item 1 from the main monitor panel. The Subsystem Information panel includes information about jobs, pipes, and the BatchPipes subsystem. **Max time** refers to the maximum amount of time jobs have been waiting, idle, or waiting-for-open.

Figure 34 shows an example of subsystem BP01 with six writers and eight readers using pipes:

- Two of the writers are in an idle condition.
- The writer that has been idle for the longest time has been idle for 15.39 seconds.
- The threshold exception field for writers identifies one idle writer as having exceeded the inactivity threshold.



```
                           BP01 Subsystem Information
Command ===>                                 Time: 21:53:25 Date:  8 NOV 1999

Press Enter to refresh data.

Subsystem:   ACTIVE       Writers:         6        Readers:          8
------------------------- ------------------------- -------------------------
Pipes:                  8   Waiting:              2   Waiting:              2
Jobs:                  14   Idle:                 2   Idle:                 2
  Waiting:              4   Wait-for-open:        2   Wait-for-open:        4


  Idle:                 4
  Wait-for-open:        6 Max time:                 Max time:
Connections:           14   Waiting:       00:01:43   Waiting:       00:15:22
Open pipe data              Idle:          00:15:39   Idle:          00:05:10
  sets:                14   Wait-for-open: 00:04:23   Wait-for-open: 00:16:12
Alloc'd pipe data         Threshold Exceptions:     Threshold Exceptions:
  sets not open:        0   Waiting:              0   Waiting:              1
                            Idle:                 1   Idle:                 0
                            Wait-for-open:        0   Wait-for-open:        2
------------------------- ------------------------- -------------------------


 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

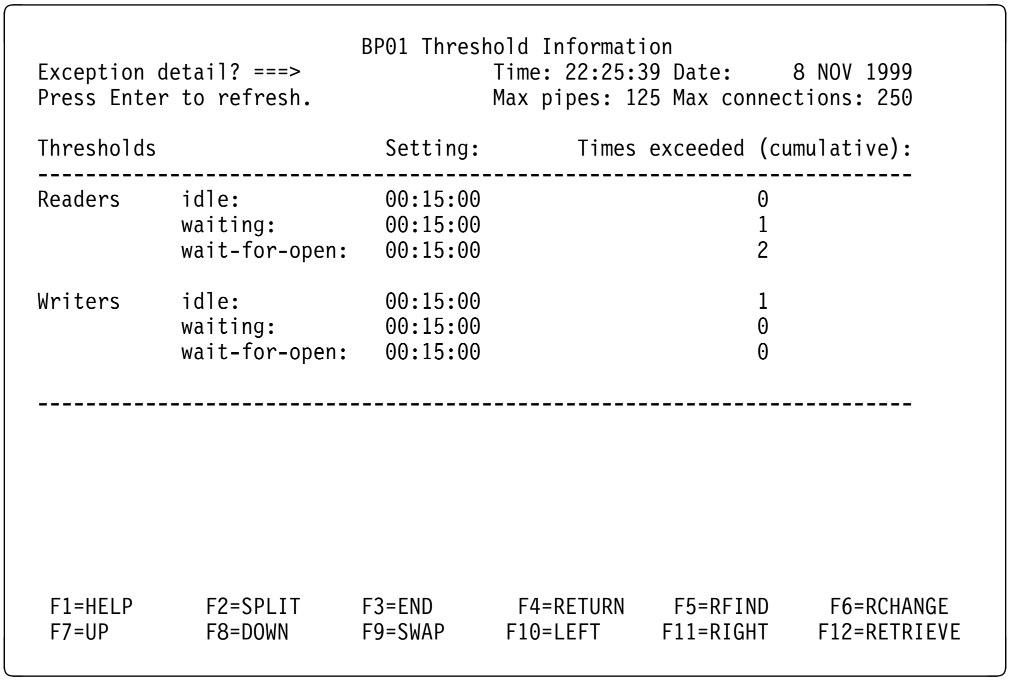
Figure 34. BatchPipes Monitor Subsystem Information Panel

Threshold Information

To obtain information about inactivity of the jobs and pipes using the subsystem, select option 2 from the main panel of the monitor. The first threshold display provides:

- Threshold settings in minutes and the cumulative count of exceptions for the life of the subsystem for each type of threshold:
  – Idle writers and readers
  – Waiting writers and readers
  – Waiting-for-open writers and readers
- BatchPipes limits on the number of pipes per subsystem
- BatchPipes limits on the number of connections per subsystem
- An input field, for you to request details about exceptions.

Figure 35 shows an example of the first threshold display, which shows that all the threshold settings are 15 minutes, the BatchPipes default.



```
                           BP01 Threshold Information
Exception detail? ===>                Time: 22:25:39 Date:     8 NOV 1999
Press Enter to refresh.               Max pipes: 125 Max connections: 250

Thresholds                   Setting:        Times exceeded (cumulative):
-------------------------------------------------------------------------
Readers     idle:            00:15:00                      0
            waiting:         00:15:00                      1
            wait-for-open:   00:15:00                      2

Writers     idle:            00:15:00                      1
            waiting:         00:15:00                      0
            wait-for-open:   00:15:00                      0

-------------------------------------------------------------------------






 F1=HELP      F2=SPLIT     F3=END        F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP      F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 35. BatchPipes Monitor Threshold Information Panel

To obtain further details about job name or wait time, select the “exception detail” field on the panel above.

From the Identify Jobs panel, you can request more information, by jobs and type of exceptions. The output displays all current threshold exceptions, according to how you request the information.

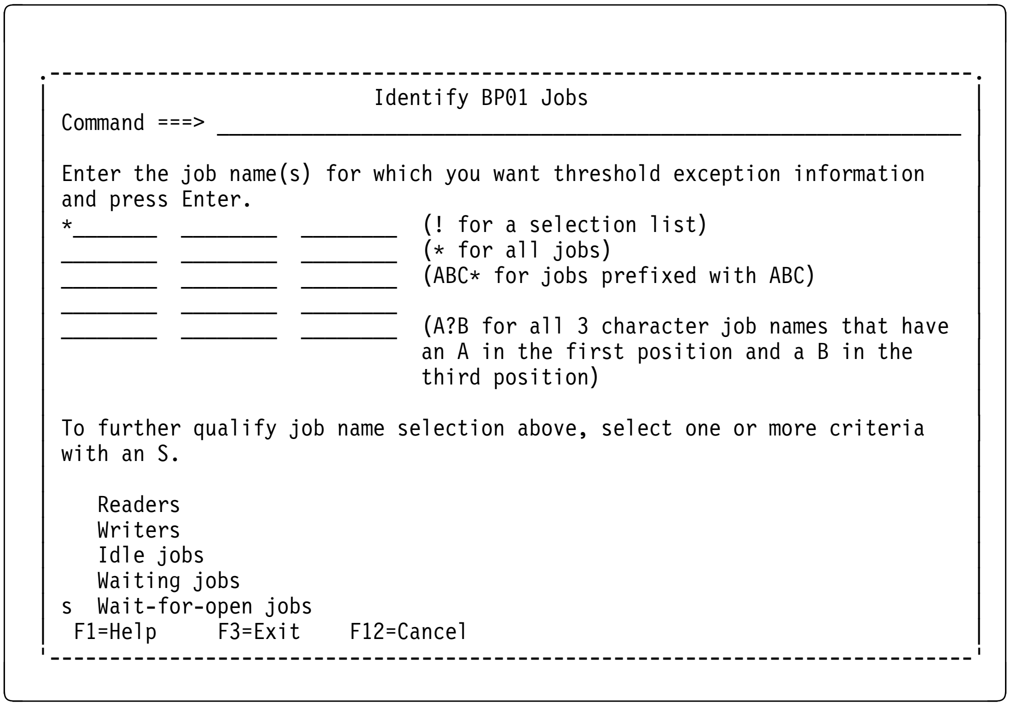
Request the information in one of these ways:

- Specify one or more job names
- Specify a job name by using a string of characters along with any combination of these two special characters:

  Question mark (“?”), which acts as a single-character **wild card**
  Asterisk (“*”), which acts as a multiple-character wild card.
- Use an asterisk to obtain a report on all jobs
- Specify an exclamation mark (“!”) to request a list of all jobs, from which you can make a selection.

On the top of the panel, select by job name; to further qualify the job name, use the bottom of the panel. If you press enter without making a selection, the monitor displays all exceptions with job names matching the criteria in the top half of the panel.

The Identify Jobs panel in Figure 36 shows how you request threshold information about all wait-for-open writers and readers that are inactivity threshold exceptions.

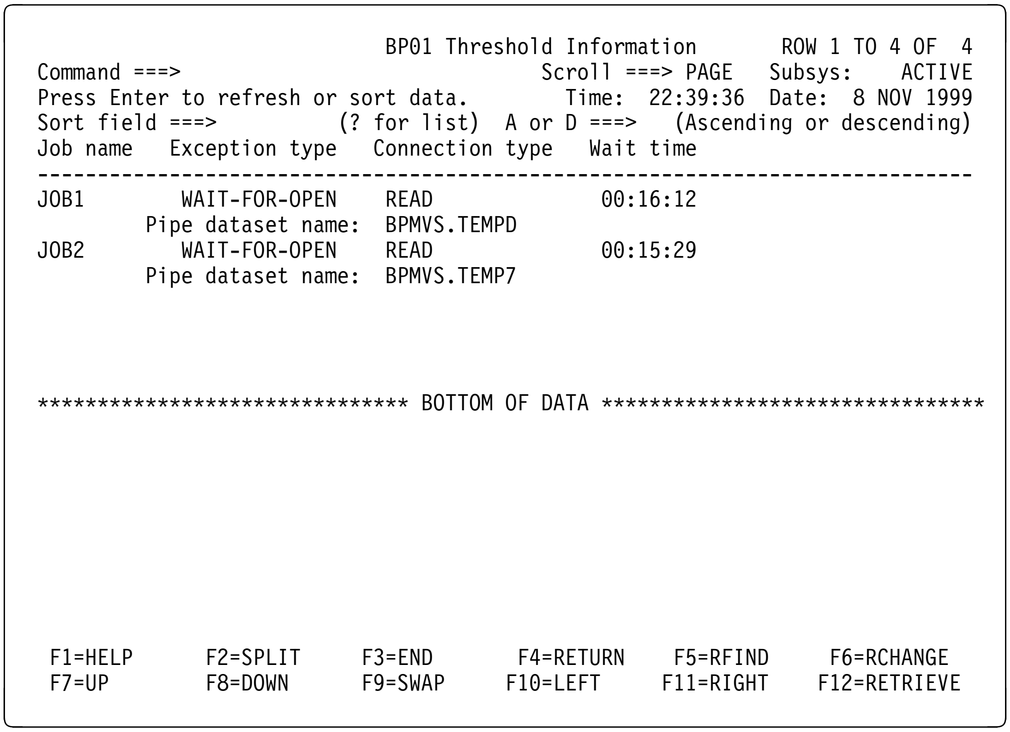


```
.------------------------------------------------------------------------------.
|                          Identify BP01 Jobs                                  |
| Command ===> ______________________________________________________________ |
|                                                                              |
| Enter the job name(s) for which you want threshold exception information     |
| and press Enter.                                                             |
| *_______  ________  ________  (! for a selection list)                       |
| ________  ________  ________  (* for all jobs)                               |
| ________  ________  ________  (ABC* for jobs prefixed with ABC)              |
| ________  ________  ________                                                 |
| ________  ________  ________  (A?B for all 3 character job names that have   |
|                               an A in the first position and a B in the      |
|                               third position)                                |
|                                                                              |
| To further qualify job name selection above, select one or more criteria     |
| with an S.                                                                   |
|                                                                              |
|    Readers                                                                   |
|    Writers                                                                   |
|    Idle jobs                                                                 |
|    Waiting jobs                                                              |
| s  Wait-for-open jobs                                                        |
|  F1=Help     F3=Exit    F12=Cancel                                           |
'------------------------------------------------------------------------------'
```

Figure 36. BatchPipes Monitor Identify Jobs Panel

In response to the selection you made on the Threshold Information panel, a summary appears of all jobs that have been in a wait-for-open state for longer periods than the inactivity threshold setting.

Figure 37 on page 100 shows an example of the Threshold Information panel, where two jobs have been waiting-to-open for longer than the wait-for-open inactivity threshold setting.



```
                             BP01 Threshold Information       ROW 1 TO 4 OF  4
Command ===>                                Scroll ===> PAGE   Subsys:    ACTIVE
Press Enter to refresh or sort data.          Time:  22:39:36  Date:  8 NOV 1999
Sort field ===>          (? for list)  A or D ===>   (Ascending or descending)
Job name   Exception type   Connection type   Wait time
--------------------------------------------------------------------------------
JOB1        WAIT-FOR-OPEN    READ              00:16:12
         Pipe dataset name:  BPMVS.TEMPD
JOB2        WAIT-FOR-OPEN    READ              00:15:29
         Pipe dataset name:  BPMVS.TEMP7




******************************* BOTTOM OF DATA ********************************








 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 37. BatchPipes Threshold Detail Panel

You can sort the information in a number of different ways, including job name and pipe name. Enter a "?" in the sort field to display a list of valid sort fields. If you do not specify the order of the sort, jobs are listed in no particular order.

Job Information

To obtain information about jobs using BatchPipes, select option 3 from the main panel of the monitor. In response, the Identify Jobs panel appears. To specify job names, specify a job name or names, job name containing any of the wild cards, or use an exclamation point to obtain a list of active jobs from which to choose.

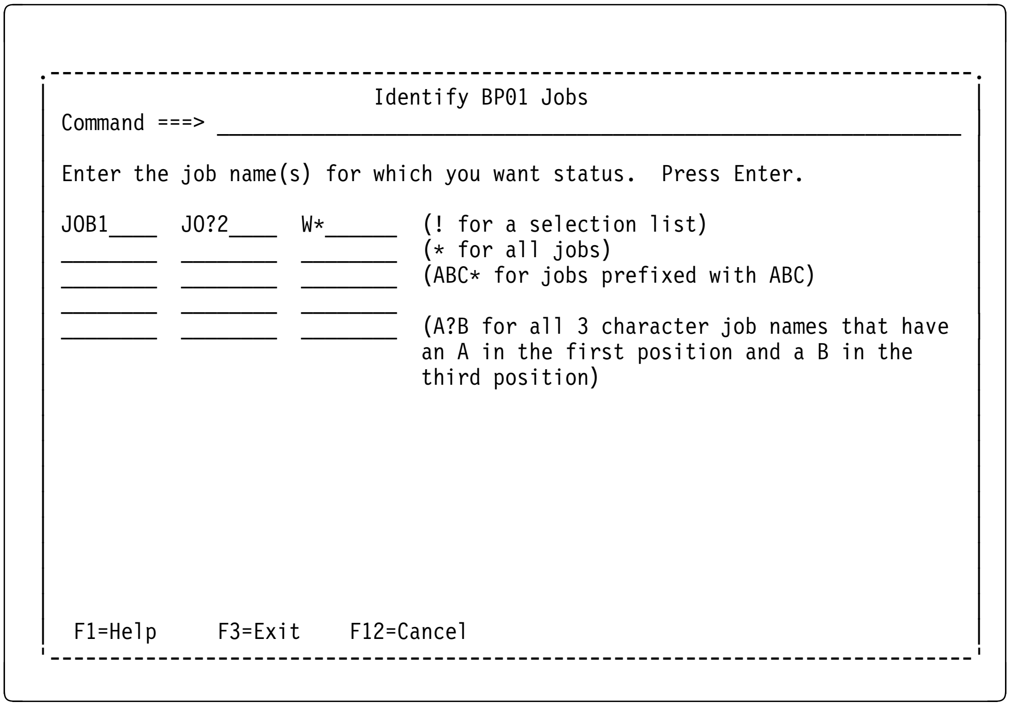
Request the information in one of these ways:

- Specify one or more pipe names
- Specify a pipe name by using a string of characters along with any combination of these two special characters:

  Question mark ("?"), which acts as a single-character **wild card**
  Asterisk ("*"), which acts as a multiple-character wild card.
- Use an asterisk to obtain a report on all pipes
- Specify an exclamation mark ("!") to request a list of all pipes, from which you can make a selection.

The Identify Jobs panel, Figure 38 shows how you request threshold information about the following jobs:

- JOB1
- Jobs with 4-character names, that begin with "JO" and end with "2"
- Jobs that begin with the letter "W."

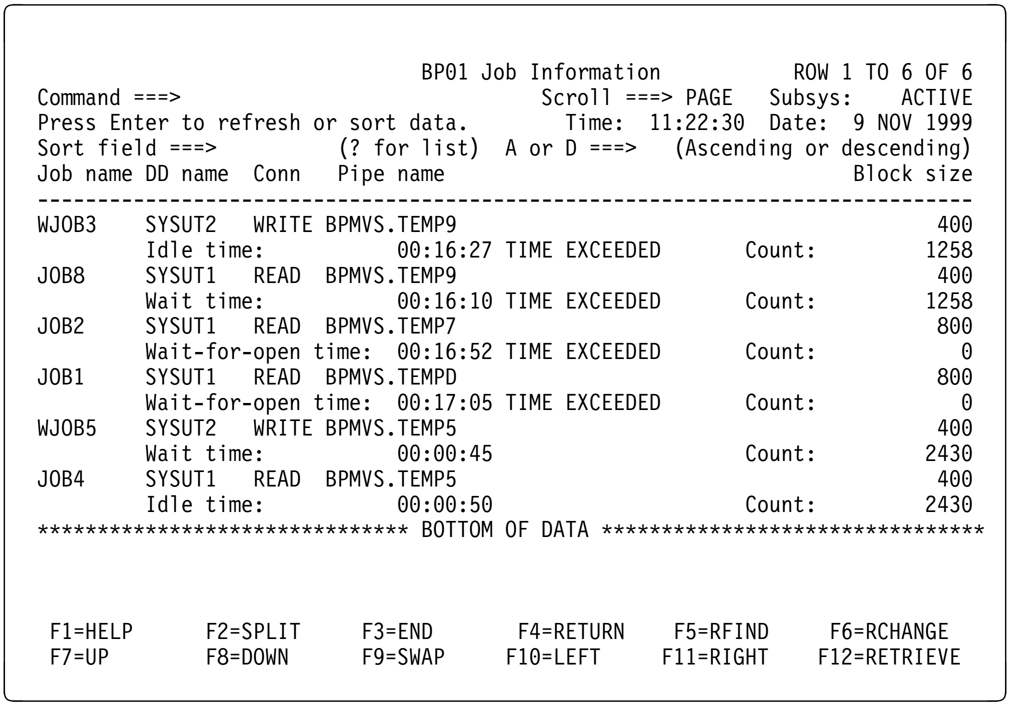


```
                          Identify BP01 Jobs
Command ===> ____________________________________________________________

Enter the job name(s) for which you want status.  Press Enter.

JOB1____  JO?2____  W*______  (! for a selection list)
________  ________  ________  (* for all jobs)
________  ________  ________  (ABC* for jobs prefixed with ABC)
________  ________  ________
________  ________  ________  (A?B for all 3 character job names that have
                              an A in the first position and a B in the
                              third position)







 F1=Help     F3=Exit    F12=Cancel
```

Figure 38. BatchPipes Monitor Job Information Identify Jobs Panel

In response to the selection you made on the Identify Jobs panel, a summary appears of all jobs that had names as described earlier. If the job is a threshold exception, it is flagged as such in the job information display.

Figure 39 on page 102 shows an example of the Job Information panel, where six jobs are displayed. The first job listed is writer WJOB3, which has been idle for 16 minutes, 27 seconds, a time that exceeds the threshold. The job has written 1258 blocks to the pipe named BPMVS.TEMP9, and the block size is 400. Because one job can be connected to more than one pipe, and a pipe can have numerous job connections, the monitor displays all connections for the selected jobs.



```
                                BP01 Job Information          ROW 1 TO 6 OF 6
Command ===>                              Scroll ===> PAGE   Subsys:    ACTIVE
Press Enter to refresh or sort data.        Time:  11:22:30  Date:  9 NOV 1999
Sort field ===>          (? for list)  A or D ===>   (Ascending or descending)
Job name DD name  Conn   Pipe name                                  Block size
------------------------------------------------------------------------------
WJOB3    SYSUT2   WRITE BPMVS.TEMP9                                        400
         Idle time:          00:16:27 TIME EXCEEDED      Count:           1258
JOB8     SYSUT1   READ  BPMVS.TEMP9                                        400
         Wait time:          00:16:10 TIME EXCEEDED      Count:           1258
JOB2     SYSUT1   READ  BPMVS.TEMP7                                        800
         Wait-for-open time:  00:16:52 TIME EXCEEDED      Count:             0
JOB1     SYSUT1   READ  BPMVS.TEMPD                                        800
         Wait-for-open time:  00:17:05 TIME EXCEEDED      Count:             0
WJOB5    SYSUT2   WRITE BPMVS.TEMP5                                        400
         Wait time:          00:00:45                    Count:           2430
JOB4     SYSUT1   READ  BPMVS.TEMP5                                        400
         Idle time:          00:00:50                    Count:           2430
******************************* BOTTOM OF DATA ********************************



 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 39. BatchPipes Job Information Panel

Pipe Information

To obtain information about pipes, select option 4 from the main panel of the monitor. In response, the Identify Pipes panel appears. Request information in one of these ways:

- Specify one or more pipe names
- Specify a pipe name by using a string of characters along with any combination of these two special characters:

  Question mark ("?"), which acts as a single-character **wild card**
  Asterisk ("*"), which acts as a multiple-character wild card.
- Use an asterisk to obtain a report on all pipes
- Specify an exclamation mark ("!") to request a list of all pipes, from which you can make a selection.

The Identify Pipes panel is similar to the Identify Jobs panel in Figure 38 on page 101; they provide the same information but in a different format. The Pipe Information panel is similar to the Job Information panel in Figure 39.

Interpreting Inactivity Threshold Messages

BatchPipes sends messages to system consoles when a job exceeds the inactivity threshold. For example, if the IDLE inactivity threshold is 20 minutes and a job named JOB8SUB that writes to DATA1.TEMP2 using the subsystem BP02 is idle for 20 minutes, the following message appears:

```
ASFP407E BP01 WRITER JOB JOB8SUB IDLE ON PIPE DATA1.TEMP2
```

Then, when activity on the pipe begins again, BatchPipes issues the message:

```
ASFO408I WRITER JOB JOB8SUB NO LONGER IDLE ON PIPE DATA1.TEMP2
```

A complete list of inactivity threshold messages (ASFP401 through ASFP412) appears with other messages in “ Appendix D: BatchPipes Messages” on page 203.

Chapter 10: Questions and Answers for the Operations Staff

This chapter lists questions that operators have asked and the answers they have received.

- **Question**: If the pipe named DAT1 is deleted, when can a pipe with that same name be created again?

  Answer: Immediately after the connections close. Pipe names must be unique within a BatchPipes subsystem.

- **Question**: What happens if a STOP command is issued after a writer and reader have both allocated data sets?

  Answer: The two jobs continue and pipe connections can be established.

- **Question**: Can a started task use a pipe?

  Answer: Yes, as long as the access method is QSAM or BSAM.

- **Question**: What happens if you don't use SUB=MSTR on the START command?

  Answer: BatchPipes terminates if JES2 or JES3 terminates.

Section IV: Information for Application Developers

Section IV is organized into the following chapters:

- **“Chapter 11: Planning Tasks for Application Developers”** on page 109 describes the planning effort for application developers.
- **“Chapter 12: Making Changes to JCL or Dynamic Allocation”** on page 117 describes the changes you make to JCL DD statements or dynamic allocation invocations.
- **“Chapter 13: Planning Recovery for Jobs Using BatchPipes”** on page 135 describes abend considerations for jobs in a pipeline.
- **“Chapter 14: Security Considerations”** on page 139 describes ways you can prevent unauthorized persons or jobs from using pipes.
- **“Chapter 15: Changing Applications to use BatchPipes”** on page 141 describes the benefits of rewriting existing applications to take advantage of BatchPipes.
- **“Chapter 16: Questions and Answers for Application Developers”** on page 149 contains questions and answers on topics that interest application developers.

The information in this section is intended for those who might perform these tasks on behalf of BatchPipes:

- Make changes to JCL statements or dynamic allocation invocations
- Make necessary changes to sort operations
- Provide recovery for applications using pipes
- Make changes to their application code to take best advantage of BatchPipes.

Chapter 11: Planning Tasks for Application Developers

This chapter describes General-use Programming Interfaces and associated guidance information.

Planning tasks for application developers include asking the following questions:

- How do jobs in a pipeline recover from abends?

  This topic is covered in "Chapter 13: Planning Recovery for Jobs Using BatchPipes" on page 135.

- When is a physical copy of the piped data still needed and how can I create that copy?

  This topic is covered in "Writing a Copy of Data to Tape or DASD" on page 148.

- How can I get the best performance from my jobs?

  This topic is covered in "Performance Considerations for Application Developers" on page 114.

- Where and how will testing of BatchPipes jobs be done?

  You should plan to run the pipeline jobs using a test-system copy of the BatchPipes subsystem.

"Checklist of Planning and Implementing Tasks" on page 27 is a checklist that one BatchPipes customer uses. It lists tasks for both system programmers and application developers. That checklist might suggest some actions for you to take.

Being responsible for the application code, you need a good understanding of some "advanced pipe concepts," such as how BatchPipes interprets the OPEN and CLOSE macros.

How BatchPipes Interprets OPEN and CLOSE Macros

Let's return to the scenario in "Description of Data Movement with BatchPipes" on page 5 and the basic pipeline of writer, reader, and one pipe. The DD statement that defines the data set tells the system that BatchPipes will be moving the data through a pipe. When the writer issues the OPEN macro, BatchPipes does not immediately open the pipe; it stays in a wait-for-open condition until the reader also opens the pipe. (A similar situation occurs if the reader opens first; the reader stays in a wait-for-open condition until the writer opens.) Then, BatchPipes builds the pipe connections. In that example, the BUFNO parameter determines that the pipe can contain as many as 7 blocks of data.

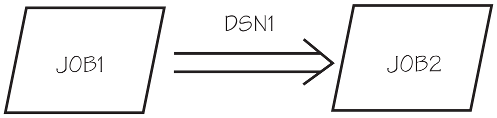
Similarly, when the writer issues the CLOSE macro, the pipe continues to empty until the last record leaves the pipe and the reader receives an end-of-file indication. When the reader issues the CLOSE macro, BatchPipes deletes the pipe.

The following considerations apply to opening and closing of data sets when the data set is a pipe. They suggest adjustments you might make to your applications.

- Until a reader and a writer have both opened the pipe, BatchPipes does not honor their write and read requests. Therefore, you might consider having the writers and readers open data sets as early as possible.
- As soon as a job finishes using a data set, it should close the data set, unless EROPT=SKP or EROPT=ACC are specified on the DCB statement.
- If, upon the opening of a data set, a job causes a X'913' abend code, check with the security administrator. The RACF authorization might:
  - Prevent you from using the BatchPipes subsystem
  - Prevent the data set from being used as a pipe by a BatchPipes subsystem.
- Each open and close request causes BatchPipes to undertake some actions. Therefore, you should avoid repeated opens and closes of a pipe by the same job.

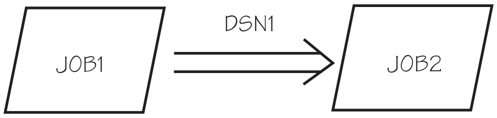
  Look at the following two scenarios to learn what happens when a reader opens and closes more than once.

  First Scenario: In the first scenario, the writer JOB1 issues an open request first followed by JOB2. Before JOB1 writes its first record, JOB2 closes. Then, when JOB1 closes, both connections end; the pipe and data are deleted. When JOB2 issues an open request again, it enters a wait-to-open state; therefore, JOB2 waits for a partner that never appears.



Sequence of events:
JOB1 OPENs for write
JOB2 OPENs for read
JOB2 CLOSEs
JOB1 writes a block
JOB1 CLOSEs.
Pipe and data are deleted
JOB2 OPENs for read,
goes into a wait, until
writer OPENs

Second Scenario: In the second scenario, the reader JOB2 issues an open request, then issues a close request. While the writer/pipe connection is still open, JOB2 again issues an open request. Therefore, the pipe continues to exist and JOB2's read request is successful.



Sequence of events:
JOB1 OPENs for write
JOB2 OPENs for read
JOB2 CLOSEs
JOB1 writes a block
JOB2 OPENs fo read
JOB1 CLOSEs
JOB2 reads from pipe
JOB2 CLOSEs

Supported Options on the OPEN Macro

The OPEN macro determines the direction of the “flow” of data through a pipe. In other words, the INPUT and OUTPUT options on the OPEN macro determine which job is the writer and which is the reader. To understand why other options on the OPEN macro are not appropriate for use with BatchPipes, consider basic differences between a data set and a pipe.

- A data set has no data flow direction. A pipe has data flow direction - from writer/pipe connection to the reader/pipe connection.
- Data can remain in a data set after it is read. However after data is read from a pipe, it is no longer available.

With BatchPipes, a job cannot use one pipe connection for both input and output within a single open. In “data set language,” a data set that is used first for input and then, without reopening, for output, cannot be a pipe. The same is true for the data set that is used first for output and then, without reopening, for input. For this reason, BatchPipes rejects jobs where the piped data sets are defined as INOUT, OUTIN, and OUTINX. When these options are specified, at the opening of the data set, the system issues abend code X'013' with reason code X'C0' and BatchPipes issues the following message to the job's joblog:

```
ASFP357I BATCHPIPES OPEN FAILURE. INOUT OR OUTIN SPECIFIED:
         JOBSTEP=STEPNAME DD=DDNAME SUBSYS=SSNAME
```

Overriding Options on the OPEN Macro

Don't automatically reject the use of BatchPipes for those data sets; the options INOUT, OUTIN, and OUTINX might not reflect a job's actual use of the data set. You can use the LABEL parameter to override the options on the OPEN macro to tell BatchPipes how your job is to use the data set:

- LABEL=(,,,OUT) indicates an output data set suitable for a write connection.
- LABEL=(,,,IN) indicates an input data set suitable for a read connection.

For example, consider a data set called DATA.TEMP that JOB2 reads. When you code the DD statement for JOB2, you specify that JOB2 is to use the pipe named DATA.TEMP, which is to be controlled by the BatchPipes subsystem named BP01.

```
//EX1  DD DSNAME=DATA.TEMP,SUBSYS=BP01,DCB=. . .
```

However, when you try to run the job, it abends with code X'013' and reason code X'C0' and message ASFP357I appears. Because you do not have access to the application source code, you cannot check (or change) the specifications on the OPEN macro. You can override the OPEN option (and avoid the abend) by using the following statement to tell BatchPipes that JOB2 is a reader from DATA.TEMP:

```
//EX1  DD DSNAME=DATA.TEMP,SUBSYS=BP01,DCB=. . . ,LABEL=(,,,IN)
```

For more information about the LABEL parameter on the DD statement, see *MVS/ESA JCL Reference*.

Considerations for Programs Written in High Level Languages

Most programs that use sequential BSAM or QSAM work with BatchPipes, requiring no changes to I/O requests and macros. Exceptions to that rule include the following, which apply to programs written in the FORTRAN and COBOL languages:

For programs written in FORTRAN:

- To indicate direction of flow use one of the following:
  - The ACTION specifier on the FORTRAN OPEN statement:

    ```
       ACTION='READ'
    or
       ACTION='WRITE'
    ```

    The ACTION statement is available in VS FORTRAN Version 2 Release 3 and later versions.
  - Include the LABEL parameter on the DD statement:

    ```
    LABEL=(,,,IN) parameter (for a reader job) and the
    LABEL=(,,,OUT) parameter (for a writer job).
    ```

    FORTRAN's support for the LABEL parameter is available in VS FORTRAN Version 2 Release 2 and later versions.
- Because BatchPipes does not support record repositioning, you cannot use the BACKSPACE, ENDFILE, and REWIND statements. In addition, you can't change the direction of data transfer while the file is open. For example, a READ statement followed by a WRITE statement is valid FORTRAN, but this sequence won't work with BatchPipes.
- The CLOSE macro causes BatchPipes to close a pipe connection.

For programs written in VS COBOL II

If you use the STAE runtime parameter (the default), an abend in one BatchPipes job is not propagated throughout the pipeline. To ensure that any abend gets propagated through the pipeline, specify the NOSTAE parameter.

If you find other exceptions, please call the IBM support center or use the readers' comment letter at the end of this book.

Considerations With Multiple Writers or Readers

Chapter 1 of this book introduces the flow of data, where the writer writes block “A” to a pipe and the reader then reads block “A,” and so forth to the next block of data. Later in that chapter, “A Pipe with Multiple Readers or Writers” on page 8 introduces the concept of “cloning” jobs to balance the pipeline and potentially improve the elapsed time. Figure 40 shows a one-to-many pipeline where two cloned readers are reading from the pipe. Because the rate of writing to the pipe (by the one writer) is closer to the rate of reading from the pipe (by the two readers), data transfers faster. In BatchPipes terms, having two readers **balances** the pipeline.

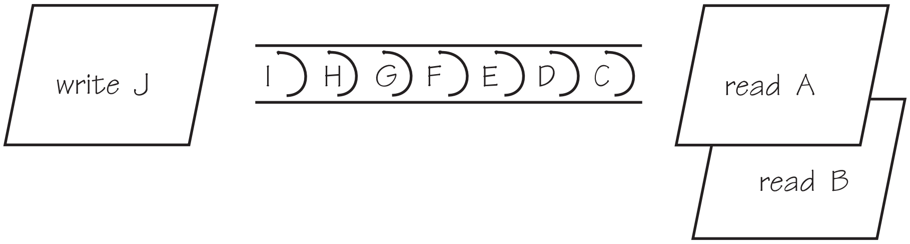


Figure 40. Example of a One-to-Many Pipeline

The two readers read from the pipe on a first come, first serve basis. For this reason, you cannot clone the following types of jobs:

- If the records are grouped in a logic unit
- If the jobs depend on the sequence of records
- If a job needs to see all the records to perform its function, for example, create a report.

Neither of the two readers reads **all** the records. The first reader to open the piped data set will most likely read more records than the second.

It is possible for two or more distinctly different writers to write to a pipe. This possibility is described in “Multiple Distinct Readers or Writers Using One Pipe” on page 145.

Similarly, a pipeline can consist of **many-to-one** stages or **many-to-many** stages. “Cloning a Writer or Reader” on page 143 tells you the steps you take to duplicate writers or readers.

Notes on Data Transfer for Multiple Writers and Readers

The following facts summarize data flow when a pipe has multiple writers or readers:

- When one writer and one reader have opened the data set that is defined to be a pipe, the pipe connections are open; once connections are open, the writer can write to the pipe. Then, the reader can read from the pipe. If a second writer or reader opens the data set, BatchPipes builds a pipe connection for that job. There are as many connections as jobs issuing OPEN requests for the pipe.
- When one writer and one reader are connected and another reader connects and issues a read, the read is satisfied with the next available record or block in the pipe. Likewise if another writer connects to an existing pipe, when the next write is issued, a record or block will transfer to the pipe.
- The order that records are written by multiple writers or read by multiple readers is not predictable. BatchPipes serves write and read requests on a first come, first served basis. See Figure 28 on page 90 for an example of a reader that read considerably fewer records from a pipe than the two other readers at the same pipeline stage.
- If a reader issues a request **and** all the writers have closed the pipe **and** the pipe is empty, the reader receives an end-of-file indication when it tries to read from a pipe that has all its writer-connections closed.

- If the readers close before the writers and the pipe is not empty, the writers will fill the pipe and wait.
- If the readers close before the writers and the pipe is empty, when the last writer issues a close, the pipe is deleted and no records are lost. This may occur by chance or as planned, with the reader expecting a given number of records and closing the pipe after receiving those records (therefore not reading to the end of the file). It is not recommended that readers close pipes after reading a specified number of records.
- The pipe and any records remaining in the pipe exist until no open connections exist. Then, BatchPipes deletes the pipe and any remaining records.
- A reader or writer might try to connect to a pipe **too late** to use the pipe — that is, after all other writer/reader partners have already used the pipe and the pipe is deleted. In this case, the late arrival waits for a partner, and for a pipe that might never be created.

Synchronizing Multiple Connections to a Pipe

When one writer and one reader are using a pipe and another reader opens the pipe data set and issues a read, the read is satisfied with the next available record or block in the pipe. Likewise, if another writer opens an existing pipe, when the next write is issued, a record or block is transferred to the pipe.

A reader or writer might try to connect to a pipe too late to use the pipe -- that is, after all other writer/reader partners have already used the pipe and the pipe is deleted. In this case, the late arrival waits for a partner, and for a pipe that might never be created.

You can avoid this situation by synchronizing the arrival of all jobs to the pipe through the OPENSYNC subparameter on the JCL SUBSYS parameter. OPENSYNC allows you to ensure that all connections are present so that no data is lost if jobs arrive late in processing. OPENSYNC specifies the number of writers and number of readers that must open the pipe data set before BatchPipes builds the pipe connections.

For a description of the subparameters on SUBSYS, see *Syntax of BatchPipes Subparameters on the SUBSYS Parameter*“Chapter 12: Making Changes to JCL or Dynamic Allocation” on page 117 .

Performance Considerations for Application Developers

You can expect your BatchPipes jobs to use the processor in much the same way they do when they are not using BatchPipes. However, all the jobs in a pipeline use the processor at the same time. Keeping that in mind, application developers can improve the performance of jobs in the following ways.

- For best performance, jobs using BatchPipes should be running on a multiprocessor. However, even on a uniprocessor, jobs get the benefit of I/O avoidance and multi-tasking, especially if the first and last jobs in the pipeline perform a lot of physical I/O operations.
- Look for cases where one job in a pipeline is slower to process records than other jobs in the pipeline. You might be able to make multiple copies of the job

to increase the speed at that stage of the pipeline. “Cloning a Writer or Reader” on page 143 contains information on how to duplicate (or clone) jobs.

- Increase the speed of processing at the “ends” of the pipeline. In some cases, the speed at which the data flows through the pipe is limited by the speed at which data flows to the first job in the pipeline and from the last job in the pipeline. One approach to increase the speed of input to the pipe is to clone the first job in the pipeline and partition the input data. Another approach is Sequential Data Striping that can increase the speed of I/O operations at pipe ends. See “Using Sequential Data Striping to Begin or End a Pipeline” for an example of Sequential Data Striping.
- Use of a pipe by multiple tasks in the same jobstep is restricted to the case where there is exactly one writer task and exactly one reader task for a given pipe. Another way of saying this is: within one jobstep there can be multiple occurrences of the same DSNAME but different DDNAMEs (presumably one DDNAME for each subtask that is using pipes.)

  Consider the following two subtasks, SUBTA, and SUBTB:

  ```
        TASK A                    TASK B
     SUBTA CSECT ,             SUBTB CSECT ,
         ...                       ...
    DCB DDNAME=DDA,           DCB DDNAME=DDB,
         RECFM=FB                  RECFM=FB
       END SUBTA                 END SUBTB
  ```

  The JCL for the two could be the following:

  ```
  //DDA  DD  DSN=PIPED.DATASET,SUBSYS=BP01,LRECL=666,RECFM=FB
  //DDB  DD  DSN=PIPED.DATASET,SUBSYS=BP01,LRECL=666,RECFM=FB
  ```

  Each subtask can open its own unique DD (file) and process using its own unique DCB. SUBTA could, for example, be the writer while SUBTB is the reader.

Performance information also appears in “Performance Tuning for BatchPipes” on page 32 in the system programmers' section.

Using Sequential Data Striping to Begin or End a Pipeline

Sequential Data Striping benefits batch jobs that spend much of their elapsed time sequentially reading or writing large DASD files. When a data set is written, Sequential Data Striping distributes the data across multiple system-managed DASD volumes. Jobs that later read the data have their read requests processed in parallel from those volumes.

Sometimes, input to the pipeline and output from the pipeline slow down the entire process. Sequential Data Striping with BatchPipes can further improve the elapsed time of the pipeline. For example, suppose Job1 reads a large data set from DASD, one record at a time. It processes each record and sends data to DASD. After the data set is closed, Job2 reads the data, one record at a time and produces a report. Figure 41 on page 116 shows Sequential Data Striping and BatchPipes together managing the application's data. Input, managed by Sequential Data Striping, comes from multiple volumes to Job1. Job1 reads and processes each record. Then, BatchPipes transfers the output to Job2, which produces the report. Without help from Sequential Data Striping, Job1's reading of data would slow down the flow of data through the pipeline.

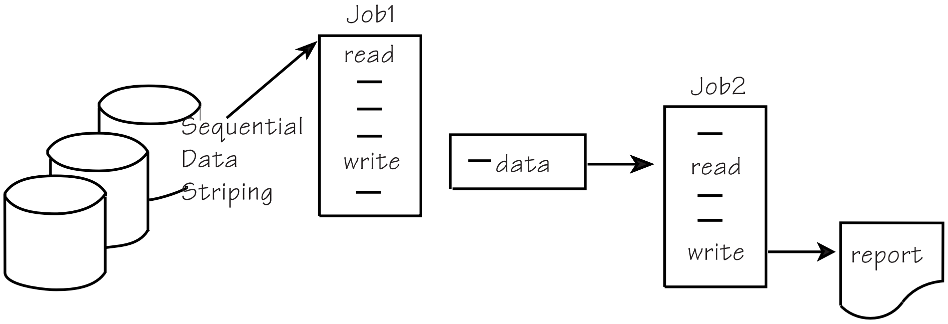


Figure 41. Using Sequential Data Striping and BatchPipes in Combination

For more information about Sequential Data Striping, see *DFSMS*/MVS: Storage Administration Guide for DFSMShsm*.

Considerations for Sort Programs in a Pipeline

A sort program in a pipeline usually increases the use of the processor. If the sort product you use accepts and uses an option that allows you to specify the “file size” (that is, the number of records or the volume of data being piped), the product can take advantage of this information and use the processor more effectively.

Other considerations about sort programs appear in “Sort Programs as Candidates” on page 15.

Chapter 12: Making Changes to JCL or Dynamic Allocation

This chapter describes the required changes that tell the system that BatchPipes rather than the I/O subsystem is to transfer data. It is assumed that at least one instance of the BatchPipes subsystem is running and that you have identified jobs that you want to use with that subsystem.

When you prepare your jobs to use BatchPipes, remember that BatchPipes acts as an “I/O surrogate”; it intercepts the application's I/O requests and directs the data through processor storage. Application programs generally continue to use the normal I/O declares, macros, or routines. However, both **writer and reader** job's JCL or dynamic allocation invocation must describe:

- The name of the BatchPipes subsystem
- The name of the data set that is to be “piped”
- The logical record length
- The record format
- The DD name.

The DD statement or a dynamic allocation invocation can provide this information. If you are using JCL, turn to “Adding the SUBSYS Parameter to the DD Statement” on page 118 and “Specifying DCB Values” on page 126 for information and examples. If you are using dynamic allocation, turn to “Using Dynamic Allocation to Specify Values” on page 130.

There might be other changes to JCL:

- For job-to-job candidates, the JCL changes are described in “Adding the SUBSYS Parameter to the DD Statement” on page 118.
- For jobstep-to-jobstep candidates, the jobsteps must become separate jobs. These JCL changes are described in “Changing the Jobstep-to-Jobstep Candidates” on page 129.
- To change jobs from using VIO to using BatchPipes, JCL changes are also required, including transforming the two jobsteps into jobs. These JCL changes are described in “Changing from VIO to BatchPipes” on page 130.
- Cloning is accomplished through a JCL change. The change can be as simple as making a second copy of the reader's JCL, keeping the same name of the data set on both DD statements, but changing the name of the job. For details, see “Cloning a Writer or Reader” on page 143.

The BatchPipes subsystem must be defined on the system that converts and executes the jobs using BatchPipes. For information on how to force conversion and execution on the system with BatchPipes installed, look up one of the following statements in the *MVS/ESA JCL Reference*:

- /*JOBPARM for JES2
- //*MAIN for JES3.

Making the JCL Changes

In all cases, the JCL must identify the pipe and the BatchPipes subsystem. On this statement, **both** writer job (or jobs) and reader job (or jobs) must identify:

- The name of the pipe (that is, the name of the data set) on the DSN keyword
- The name of the BatchPipes subsystem on the SUBSYS keyword
- The DCB attributes RECFM and LRECL.

BatchPipes uses values on the DSN and SUBSYS keywords to associate the writer (or writers) with the reader (or readers) that use the same pipe.

Adding the SUBSYS Parameter to the DD Statement

The SUBSYS parameter offers subparameters that require you to make decisions. This section suggests questions you need to ask about your jobs and describes subparameters that apply.

- Do you want to use a BatchPipeWorks fitting to change the flow of data through the pipe? If so, use the FIT or FITDD subparameters.

- Do you want a specified number of writers and readers ready to use the pipe before BatchPipes builds connections? If so, use the OPENSYNC subparameter.

- Do you want the pipe connection to open and begin using the pipe, even before a partner has opened? If so, use the OPENNOW subparameter.

- Do you want the writers to close without sending an EOF indication to reader partners? If so, use the NOEOF subparameter. You would us this if you want your readers to remain open after all writers close.

- Do you want to prevent your readers from closing the pipe data set until they receive an EOF indication? If so, use the EOFREQUIRED=YES subparameter. This option ensures that a reader receives an EOF indication before it closes.

- Do you want all writers and readers using the pipe to close the pip data set before BatchPipes deletes the pipe connections and allows the jobs to continue processing? If so, use the CLOSESYNC subparameter. By synchronizing the closings of writers and readers, BatchPipes can detect abnormally-ending jobs and propagate the I/O errors before the closes occur.

- Does the reading program read all of the data before closing the input file? If not, use the EOFREQUIRED=NO option on the reading job, and ERC=DUMMY on the writing job.

- Do you want to limit error propagations so that a permanent copy of the data is written if downstream errors occur? If so, use the DCB option EROPT=SKP to protect the writing job.

- Do you want all jobs to either fail together or be successful together (that is, if you do not want some jobs to fail while others end successfully)? If so, use the TERMSYNC option.

- Do you anticipate abnormal wait conditions for a specific pipe? If so, use pipe-level thresholds to control the threshold messages to be issued.

Specifying the Name of the Subsystem and the Pipe

To tell the system that the writer and reader are using a pipe rather than tape or DASD, code a DD statement that identifies the BatchPipes subsystem and the pipe. The JCL would look like this:

```
//ddname DD DSN=dsname,SUBSYS=bp-subsystem-name,
             DCB=(LRECL=www,RECFM=xx)
          or
//ddname DD DSN=dsname,SUBSYS=bp-subsystem-name,LRECL=xxx,RECFM=xx
```

Where:

dsname

The name of the pipe. **IBM recommends** that you establish a naming convention for the pipes. Having a naming convention helps the operations staff understand BatchPipes messages and helps the security administrator restrict the use of pipes. You can use data set names that reflect the type of processing that the pipe handles. For example, by assigning the name BILLING as the high-level qualifier for the pipes in a pipeline, you can specify:

bp-cmd-prefix STATUS,P=BILLING.*

to obtain information about the pipeline that is associated with the billing process.

bp-subsystem-name

The name of the BatchPipes subsystem. This name must match a BatchPipes entry in IEFSSNxx member of PARMLIB. Check with system programming.

Syntax of BatchPipes Subparameters on the SUBSYS Parameter

The complete syntax of BatchPipes subparameters on the JCL SUBSYS parameter follows. The subparameters in the diagram have uppercase and lowercase letters that indicate the shorthand version. For help reading the syntax representation, called “railroad tracks,” see “Appendix F: Understanding Syntax Diagrams” on page 287.

The syntax of the SUBSYS parameter is as follows:

SUBSYS Parameter Syntax

```
►►──SUBSYS=──┬─ssnm──────────────────────────────────────────────────────────┬──────────►◄
             │         ┌─,─────────────────────────────────────────────┐     │
             └─(──ssnm─▼─┬─'AllocSync=n'─────────────────────────────┬─┴──)─┘
                         ├─AllocNow──────────────────────────────────┤
                         ├─'Fit=fitting specification'───────────────┤
                         ├─'FITDD=ddname'─┬─────────────────────────┬┤
                         │                └─'CMT=comment-delimiter'─┘│
                         │                 ┌─,─────┐                 │
                         ├─'OpenSync=──(───▼─┬─R=r─┬┴──)──'──────────┤
                         │                   └─W=w─┘                 │
                         ├─OpenNow───────────────────────────────────┤
                         ├─NOEOF─────────────────────────────────────┤
                         ├─'EOFRequired=──┬─NO──┬──'─────────────────┤
                         │                └─YES─┘                    │
                         ├─CloseSync─────────────────────────────────┤
                         ├─TermSync──────────────────────────────────┤
                         ├─'TermSync=cc'─────────────────────────────┤
                         ├─'ERC=──┬─Continue─┬──'────────────────────┤
                         │        └─DUMMY────┘                       │
                         ├─'WaitAlloc=──┬─nnnn─┬──'──────────────────┤
                         │              └─OFF──┘                     │
                         ├─'WaitOpen=──┬─nnnn─┬──'───────────────────┤
                         │             └─OFF──┘                      │
                         ├─'Idle=──┬─nnnn─┬──'───────────────────────┤
                         │         └─OFF──┘                          │
                         ├─'Wait=──┬─nnnn─┬──'───────────────────────┤
                         │         └─OFF──┘                          │
                         ├─'WaitEOF=──┬─nnnn─┬──'────────────────────┤
                         │            └─OFF──┘                       │
                         ├─'WaitClose=──┬─nnnn─┬──'──────────────────┤
                         │              └─OFF──┘                     │
                         ├─'WaitTerm=──┬─nnnn─┬──'───────────────────┤
                         │             └─OFF──┘                      │
                         ├─'PipeDepth=nnn'───────────────────────────┤
                         └─'ERRPROP=──┬─CANCEL─┬──'──────────────────┘
                                      ├─ABEND──┤
                                      ├─CONT───┤
                                      └─DUMMY──┘
```

The descriptions of the subparameters on the SUBSYS parameter on the DD JCL statement are as follows:

Operands

ssnm
The name of the BatchPipes subsystem. This name must match a BatchPipes entry in IEFSSNxx member of PARMLIB. Check with the system programmers at your installation.

ALLOCSYNC

Synchronizes a particular pipeline at pipe data set allocation. Use ALLOCSYNC to specify the minimum number of jobs that must allocate a particular pipe data set before allocation can complete for all jobs in a pipe.

Because BatchPipes error detection and propagation does not start until allocation time, using ALLOCSYNC ensures that jobs that allocate a pipe data set at widely varying times are covered by BatchPipes error detection and propagation.

Use ALLOCSYNC when the period over which different jobs allocate a particular pipe data set is fairly long.

ALLOCNOW

Allows a job that might otherwise wait during ALLOCSYNC processing to complete allocation. The job can specify ALLOCSYNC, and count toward the ALLOCSYNC value, but the job itself does not wait during allocation for the ALLOCSYNC value to be met.

ALLOCNOW allows jobs that hold vital resources to avoid deadlocks during allocation time. These jobs can complete allocation without waiting so that their resources can be released.

FIT=fitting specification

Identifies the fitting specification -- the BatchPipeWorks stages that comprise the fitting.

FITDD=ddname

Specifies the ddname that contains the BatchPipeWorks stages that comprise the fitting.
CMT=comment-delimiter
Specifies the character that you use as a comment delimiter. This allows you to have comments accompany your BatchPipeWorks stages. CMT is helpful when sequence numbers exist in the data set associated with the fitting. A comment on every line (even if it is null) strips the sequence numbers

OPENSYNC

Specifies the number of writers and readers required to open the pipe data set before BatchPipes builds the connections and allows the data to transfer through. Unless the OPENNOW subparameter is specified, BatchPipes holds all readers and writers until the number specified in R=r or W=w is reached. Once the number of writers or readers specified on the OPENSYNC parameter is reached, BatchPipes builds the connections. All jobs using the pipe must specify the same values for OPENSYNC. **R=r**
The number of readers required to have opened before BatchPipes builds pipe connections. The default value is 1. The allowed range is: R=r + W=w <=250. **W=w**
The number of writers required to have opened before BatchPipes builds pipe connections. The default value is 1. The allowed range is: W=w + R=r <=250.

OPENNOW

Specifies that BatchPipes is to build the pipe connection as soon as the writer or reader opens the pipe data set. Use OPENNOW to control the building of pipe connections when your writer or reader opens the pipe data set early in its processing, but does not use the pipe until much later.

This subparameter is required for half-pipe fittings. For information about half-pipe fittings.

You can use the subparameters OPENNOW and OPENSYNC together. The OPENNOW subparameter affects the writer or reader connection for which it is specified. When you specify OPENNOW and OPENSYNC(R=r,W=w), BatchPipes builds that connection:

- For a writer, when the w value is satisfied
- For a reader, when the r value is satisfied.

NOEOF

Applies to writers only; specifies that BatchPipes does not present an EOF indication to the reader when the writer closes the pipe data set. BatchPipes deletes the writer connection but not the reader connection. Withholding the EOF indication means the reader connection can remain open to serve writers that open the pipe data set later.

EOFREQUIRED=NO or YES

Specifies what happens if a reader closes a pipe data set before it receives an EOF indication. If you specify YES and a reader job closes before it receives the EOF indication, the reader abends and all writers and readers using the pipe receive an I/O error.

EOFREQUIRED=NO means the reader can close a pipe data set before it receives an EOF indication.

The default setting for EOFREQUIRED is the setting established in ASFPBPxx parm lib member.

CLOSESYNC

Specifies that all writers and readers using the pipe must close the pipe data set before BatchPipes deletes the pipe connections and allows the jobs to continue processing.

CLOSESYNC settings must be the same for all jobs using the pipe. If you do not specify CLOSESYNC, BatchPipes deletes pipe connections as the jobs close the pipe data set.

TERMSYNC

Synchronizes a particular pipeline at jobstep termination. When you specify TERMSYNC for a pipeline, each jobstep that is connected to a pipe on the pipeline waits in normal termination until its partner jobs in the pipeline arrive at termination.

TERMSYNC provides for more complete error propagation. A jobstep that completes successfully, but early, is not allowed to end until all other jobsteps on the pipeline have also ended successfully. As a result, jobsteps that end early continue to receive propagated errors from pipeline jobsteps that abend, or (optionally) end normally with an unexpected condition code.

Jobs on a TERMSYNC pipeline can specify either the TS format or the ‘TS=cc’ format of TERMSYNC. The ‘TS=cc’ format of TERMSYNC causes the system to propagate an error if the jobstep condition code is greater than or equal to cc. The condition codes specified with the ‘TS=cc’ option may vary from job to job on the pipeline.

ERC

Specifies the disposition of writer connections to a pipe if all readers of the pipe close the pipe before the writer completes.

ERC=CONT
(the default) indicates that on early reader close, processing continues as normal. I/O continues to place data in the pipe.

ERC=DUMMY
indicates that on early reader close, processing continue with the pipe becoming DUMMY. Subsequent I/O requests to this DD statement do not result in data flowing into the pipe. However, data continues to flow into a fitting if one is coded on this DD statement.

IBM recommends that you specify EOFRequired=NO for readers that read only a portion of the pipe data set.

Specifying EROPT=ACC/SKP/ABE takes precedence over ERC=DUMMY. In case, if I/O errors occur, the processing specified on the EROPT keyword determines the processing actions.

IDLE=nnnn or OFF

Specifies the amount of time a writer or reader can remain idle before BatchPipes issues a message. IDLE=nnnn indicates the number of minutes, ranging from 0 to 1440.

The default setting for EOFREQUIRED is the setting established in ASFPBPxx parm lib member. The value specified for the IDLE=nnnn option overrides the setting of the IDLE=nnnn setting in the ASFPBPxx parm lib member.

It is recommended that you do not set a value of 0 for the idle threshold.

If you specify IDLE=OFF, BatchPipes does not monitor the idle state

WAIT=nnnn or OFF
Specifies the amount of time a writer or reader can remain in a wait state before BatchPipes issues a message. WAIT=nnnn indicates the number of minutes, ranging from 0 to 1440.

The default setting for EOFREQUIRED is the setting established in ASFPBPxx parm lib member. The value specified for the WAIT=nnnn option overrides the setting of the WAIT=nnnn setting in the ASFPBPxx parm lib member.

It is recommended that you do not set a value of 0 for the wait threshold.

If you specify WAIT=OFF, BatchPipes does not monitor the wait state

WAITALLOC=nnnn or OFF
Specifies the amount of time a writer or reader can remain in a wait-for-allocation-sync state before BatchPipes issues a message. WAITALLOC=nnnn indicates the number of minutes, ranging from 0 to 1440.

The default setting for EOFREQUIRED is the setting established in ASFPBPxx parm lib member.The value specified for the WAITALLOC=nnnn option overrides the setting of the WAITALLOC=nnnn setting in the ASFPBPxx parm lib member.

If you specify WAITALLOC=OFF, BatchPipes does not monitor the wait-for-allocation state.

WAITOPEN=nnnn or OFF
Specifies the amount of time a writer or reader can remain in a wait-for-open state before BatchPipes issues a message. WAITOPEN=nnnn indicates the number of minutes, ranging from 0 to 1440.

The default setting for EOFREQUIRED is the setting established in ASFPBPxx parm lib member. The value specified for the WAITOPEN=nnnn option overrides the setting of the WAITOPEN=nnnn setting in the ASFPBPxx parm lib member.

If you specify WAITOPEN=OFF, BatchPipes does not monitor the wait-for-open state.

WAITCLOSE=nnnn or OFF
Specifies the amount of time a writer or reader can remain in a wait-for-close state before BatchPipes issues a message. WAITCLOSE=nnnn indicates the number of minutes, ranging from 0 to 1440.

The default setting for EOFREQUIRED is the setting established in ASFPBPxx parm lib member. The value specified for the WAITCLOSE=nnnn option overrides the setting of the WAITCLOSE=nnnn setting in the ASFPBPxx parm lib member.

If you specify WAITCLOSE=OFF, BatchPipes does not monitor the wait-for-close state.

WAITEOF=nnnn or OFF
Defines the amount of time a reader can remain in a wait-for-EOF state before BatchPipes sends a message. WAITEOF=nnnn indicates the number of minutes, ranging from 0 to 1440.

The default setting for EOFREQUIRED is the setting established in ASFPBPxx parm lib member. The value specified for the WAITOF=nnnn option overrides the setting of the WAITOF=nnnn setting in the ASFPBPxx parm lib member.

If you specify WAITEOF=OFF, BatchPipes does not monitor the wait-for-EOF state.

WAITTERM=nnnn or OFF

Specifies the amount of time a writer or reader can remain in a wait-for-termination sync state before BatchPipes issues a message WAITTERM=nnnn indicates the number of minutes, ranging from 0 to 1440.

The default setting for EOFREQUIRED is the setting established in ASFPBPxx parm lib member. The value specified for the WAITTERM=nnnn option overrides the setting of the WAITTERM=nnnn setting in the ASFPBPxx parm lib member.

If you specify WAITTERM=OFF, BatchPipes does not monitor the wait-for-termination state.

PIPEDEPTH=nnn

Used to control the number of buffers in storage (or in the coupling facility for cross system pipes) that will be used to hold data. If PIPEDEPTH is specified, the BUFNO DCB parameter is not used to specify pipedepth.

nnn can be a value from 1 to 32768.

nnn specifies the number of blocks that can be in the pipe at any given time before the pipe becomes “full.” If nnnnn is greater than the value specified for MAXBUFNO in parmlib, the user for the job must have authority to use values greater than MAXBUFNO. If the authority check fails, MAXBUFNO will be used and a warning message is issued.

The use of large values for PIPEDEPTH enables your application to accumulate more information in storage (or the coupling facility), or between the parallel applications. This can be helpful in smoothing out periodic wait periods where the writing application is forced to wait because the pipe temporarily becomes full because the reading application is temporarily delayed in reading data from the pipe.

For a cross-system pipe, it is possible that nnnnn buffers will not be available in the coupling facility. In this case, as many available buffers are used as possible (up to nnnnn). The number of buffers depends on how much space in the coupling facility is allocated to the BatchPipes structure.

ERRPROP=CANCEL or ABEND or CONT or DUMMY

This option is an override of the DCB-specified parm EROPT and is used to control the processing of a job in case that that job's partner has abended (error propagation). It can be used when either the access to the EROPT not sufficient and cannot be changed (for example if the program being run has hardcoded EROPT=SKP but the required action is really what is provided by EROPT=ABE) or, the special case, if the program/job that is the target of an error propagation “CANCE” request can not tolerate or support the CANCEL command processing (in which case the user can request that the target program get ABENDed with a `x'BC6'` instead of just a CANCEL command).

ERRPROP=CANCEL

requests that the targeted job be CANCELled in the case of an error propagation request (this is similar to EROPT=ABE)

ERRPROP=ABEND

requests that the targeted job be ABENDED with a `x'BC6'/x'E68710E4'` instead of being CANCELled (this is a special variation of EROPT=ABE)

Note: This option should ONLY be used in cases where it is really needed since most recovery processing is set-up to handle the CANCEL case ‘cleanly’ and this ABEND code is not an expected one and can result in excess dumps being taken by other products' recovery routines.

ERRPROP=CONT

requests that the targeted job NOT be ended in the case of an error propagation request and that the I/O to the pipe is maintained (this is similar to EROPT=ACC)

Note: This option, its EROPT equivalent, should be used with care; because when the partner job ABENDed any data in its buffers was lost and will not be processed by any other job connected to be pipe. Also, this option should really only be used when there are ‘other’ connections to the pipe of the same type (reader/writer) as the targeted job otherwise a pipe hang situation could develop in the case of an error propagation since there wouldn't be a completed pipe with a full set of connections (a reader and a writer).

ERRPROP=DUMMY
requests that the targeted job NOT be ended in the case of an error propagation request and that ALL I/O to the pipe be stopped BUT the ABENDed partner job has been ended. (this is similar to EROPT=SKP)

Examples of Using Subparameters on the SUBSYS Parameter: Example 1: To specify that two readers must open the pipe data set before BatchPipes builds the reader connections, specify the following:

```
//DD1  DD  DSN=...,SUBSYS=(BP01,'OPENSYNC=(R=2)'),...
```

Suppose one reader and two writers open the pipe data set. BatchPipes does not build the connections yet. Only when the second reader appears does BatchPipes build connections and allow data to transfer through the pipe.

Example 2: To specify that you want five reader and three writers to open the pipe data set before BatchPipes builds the pipe connections, specify the following:

```
//DD1  DD  DSN=...,SUBSYS=(BP01,'OPENSYNC=(R=5,W=3)'),...
```

Example 3: Example 3 builds on Example 2, where five readers and three writers must open the pipe data set before BatchPipes builds the connections. Suppose you want a writer's connection to open as soon as all three writers have opened the pipe data set, without regard to when the readers open. That writer would specify:

```
//DD1  DD  DSN=...,SUBSYS=(BP01,'OPENSYNC=(R=5,W=3),OPENNOW'),...
```

Because OPENNOW is specified for the writer, when the third writer open the pipe data set, BatchPipes builds the writer's connection, but no other connections.

Example 4: Suppose writer/reader partners WRITEP and READP use two pipes PIPEA and PIPEB. The writer WRITEP opens PIPEA first and PIPEB second. The reader READP opens PIPEB first and PIPEA second. To avoid having both jobs remain in a wait-for-open state, waiting for BatchPipes to build pipe connections, use OPENNOW on the JCL SUBSYS parameter for READP (the reader that issues the open for PIPEB first), as follows:

```
//DD1   DD  DSN=PIPEB,SUBSYS=(BP01,'OPENNOW'),...
```

With OPENNOW specified, the order in which the two jobs open the two data sets does not matter.

Example 5: JOBB has a BatchPipeWorks half-pipe fitting that selects certain records to be written to DASD. Use OPENNOW, as follows:

```
//DDOUT  DD  DSN=PIPE3,SUBSYS=(BP01,'OPENNOW'),...
```

Using BatchPipeWorks Fittings Outside a Pipeline: Through BatchPipes, you can use the BatchPipeWorks stages with jobs that are not in a pipeline. For example, consider the non-BatchPipes job that processes only a subset of the records that it reads -- or would like the fields in the records to be rearranged. You can have a BatchPipeWorks fitting associated with that job even though it is neither a BatchPipes reader or a writer. This kind of fitting is called a **half-pipe fitting**, a fitting that is not in a pipeline. The job to which a half-pipe fitting is associated does not have a partner.

Using Unnamed Temporary Data Sets: Under limited circumstances, an unnamed temporary data sets (data sets that you define using an ampersand-qualified data set name, such as DSN=&temp01) can be a pipe. Generally, you cannot use unnamed temporary data sets with BatchPipes. When you use an ampersand-qualified data set name on the SUBSYS DD statement, the system generates a pipe name that is unique across jobs and jobsteps. This means that multiple programs (in different jobs or jobsteps) cannot access the same pipe with that system-generated name. However, if the reader and writer of a pipe are both in the same jobstep, they **can** use an ampersand-qualified data set name. The JCL for the writer and reader contains DD statements for the pipe: one open for output and the other for input.

Specifying DCB Values

Certain JCL values that define the data set take on new meaning with BatchPipes, as described in the following sections. The first and most important rule is for RECFM and LRECL values:

Important rule for RECFM and LRECL values

Both the writer and reader must have the same values for RECFM and LRECL. If the values don't match, the second job to issue the OPEN fails and the first job to issue the OPEN remains in a wait-to-open state.

Specifying RECFM and LRECL Values

The RECFM values for writer and reader must be the same. For example, the writer's value cannot be FB if the reader's value is VB. Acceptable values on RECFM are F, FB, V, and VB.

Another important rule is: a writer or reader can't close a pipe and then open it again with RECFM and LRECL values that are different from the original allocation. Values on other DD parameters can change after a close and a reopen of a pipe; for example, the identity of a job as a writer or reader can change.

Specifying the BLKSIZE Value

BatchPipes uses the BLKSIZE value to determine how many records to move to and from the pipe at one time. You can specify a BLKSIZE value that tells BatchPipes to move the records one at a time, an inefficient data movement. Or, you can request that BatchPipes transfer multiple records' worth of data through the pipe. The larger the BLKSIZE value, the more efficiently BatchPipes moves the data. For example, compare two BLKSIZE values for COBOL applications that write fixed length 80-byte records:

If the BLKSIZE is equal to the length of a record, the data movement from the writer to the pipe occurs with each WRITE instruction.

If the BLKSIZE is equal to 32000 bytes, the data movement occurs with each 400th WRITE instruction.

IBM recommends that you omit the BLKSIZE parameter, which means:

- For fixed blocked records, the system uses the largest blocksize that is an integer multiple of the LRECL but less than 32670
- For variable blocked records, the system uses the value 32760.

Notes on specifying BLKSIZE:

- If you specify a BLKSIZE value for unblocked records, BatchPipes ignores it.
- BLKSIZE=0 gives you the maximum block size.
- BLKSIZE values for a writer and reader can differ as long as the reader's value is greater than the writer's.

- If you make a change to the BLKSIZE specification, you might need to change the value of the REGION parameter to avoid running out if virtual storage.

Specifying BUFNO Values

The BUFNO value determines

- The depth of the pipe

  The value is the depth. For example, the scenario in "Chapter 1: Introduction to BatchPipes" on page 3 shows a pipe with depth of 7, reflecting the value 7 on the BUFNO parameter.

- The size of the pipe.

  Size refers to the maximum amount of data a pipe can hold. The size of a pipe depends on values on the BUFNO and BLKSIZE parameters.

IBM recommends that you take the default: BUFNO=7. A larger number increases the use of virtual storage for the BatchPipes subsystem.

Other JCL Considerations

Some parameters on the DD statement have no meaning to BatchPipes and are ignored. For example, BatchPipes is not interested in the DISP parameter. You might have values on DISP that trigger certain events for your programs or vendor products.

If your application issues the DFSMS/MVS RDJFCB macro and depends on information in the job file control block (JFCB), you might need to code DSORG=PS on the DD statement that defines the data set that is to be the pipe.

```
DCB=(DSORG=PS, ...).
```

Example of Running BatchPipes with Two Existing Jobs

The best way to describe the required JCL change is to show an example.

- Job1 reads a record, copies it using IEBGENER, writing the record to DASD. It processes each of ten records in this way.
- Job2 reads the records one at a time, again using IEBGENER, printing each record.

Before BatchPipes

The JCL for the job NBP01 looks like the following, where the SYSUT2 DD statement describes the output data set HDDATA.TEMP that will reside on DASD.

```
//NBP01  JOB TIME=NOLIMIT,
//            REGION=2000K,MSGCLASS=A,
//            MSGLEVEL=(1,1)
//COPY1 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT2 DD DSN=HDDATA.TEMP,DCB=(LRECL=80,BLKSIZE=4000,RECFM=FB,
//   DSORG=PS),SPACE=(TRK,(1,1)),DISP=(NEW,CATLG)
//SYSUT1 DD *
RECORD 001
 .
 .
RECORD 010
```

The JCL for the job NBP02 looks like the following, where the SYSUT1 DD statement describes the input data set HDDATA.TEMP that resides on DASD.

```
//NBP02  JOB TIME=NOLIMIT,
//            REGION=2000K,MSGCLASS=A,
//            MSGLEVEL=(1,1)
//COPY2 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT1 DD DSN=HDDATA.TEMP,DISP=(OLD,DELETE)
//SYSUT2 DD SYSOUT=*
```

With BatchPipes

Let's change this JCL so that the two jobs are writer/reader partners using a pipe, rather than DASD. In this case, the name of the pipe is HDDATA.TEMP. The two jobs must be scheduled to run concurrently.

The JCL for the job WITHBP1 looks like the following where WITHBP1 writes to the pipe named HDDATA.TEMP associated with the BatchPipes subsystem named BP01.

```
//WITHBP1  JOB TIME=NOLIMIT,
//            REGION=2000K,MSGCLASS=A,
//            MSGLEVEL=(1,1)
//COPY1 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT2 DD DSN=HDDATA.TEMP,SUBSYS=BP01,
//          DCB=(LRECL=80,RECFM=FB)
//SYSUT1 DD *
RECORD 001
 .
 .
RECORD 010
```

The SYSUT2 DD statement for the job WITHBP1 names the pipe and the BatchPipes subsystem.

The JCL for the job WITHBP2 includes a SYSUT1 DD statement that also names the pipe and the BatchPipes subsystem:

```
//WITHBP2  JOB TIME=NOLIMIT,
//            REGION=2000K,MSGCLASS=A,
//            MSGLEVEL=(1,1)
//COPY2 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT1 DD DSN=HDDATA.TEMP,SUBSYS=BP01,
//   DCB=(LRECL=80,RECFM=FB)
//SYSUT2 DD SYSOUT=*
```

Changing the Jobstep-to-Jobstep Candidates

So far, this chapter has described the changes you make when writer and reader are separate jobs. When you analyze your batch applications looking for candidates, don't ignore the potential jobstep-to-jobstep writers and readers, where writer and reader are in the same job. Through a straightforward JCL change, you can change the two jobsteps into two jobs and get the benefits of BatchPipes After you change the JCL, update the scheduling package to reflect the two new jobs.

For an example of making a multiple-jobstep job into multiple jobs, look at the following two-step job.

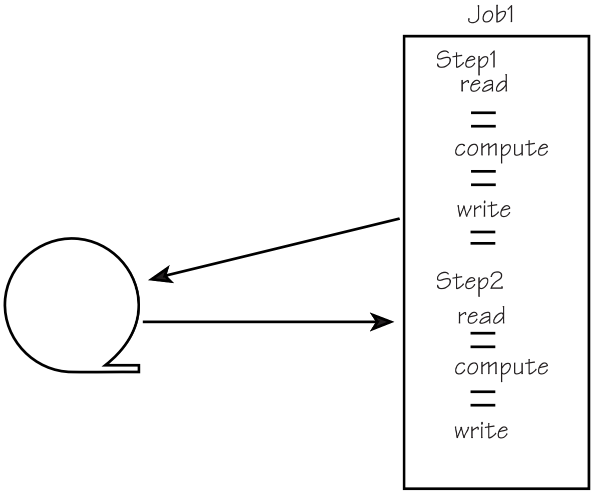


Figure 42. Jobstep-to-Jobstep Candidates

With some changes to the JCL for Job1, you can pipe the data between JobA (formerly Step1) and JobB (formerly Step2).

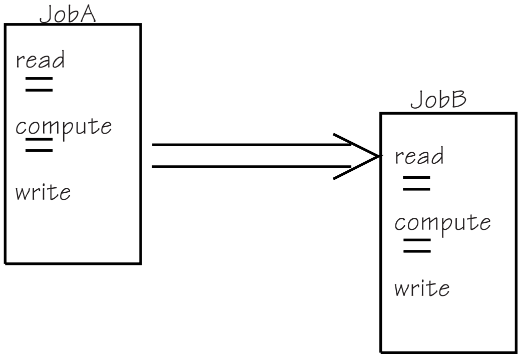


Figure 43. Change Jobstep Candidates into Jobs

Before changing jobsteps into jobs, read carefully the basic criteria described in “Criteria for Good Candidates” on page 13. Make sure:

- No job later in the jobstream needs the “piped” data, even for application restarts. If you “pipe” data between two jobsteps, that data is not available for any job that runs later in the jobstream.
- Non-piped data sets are **not** allocated in one step and deleted in a later step. For example, consider two jobsteps named STEP1 and STEP2:

  ```
  STEP1 uses data that is defined as DSN=MAJ.DATA,...DISP=(,PASS)...
  STEP2 uses data that is defined as DSN=MAJ.DATA,...DISP=(OLD,DELETE)...
  ```

  If you changed the two steps into jobs and ran them in parallel, deletion of the data set might occur before the first job (the old STEP1) finished using the data set named MAJ.DATA.
- You have enough initiators to handle the increased number of jobs running at one time
- All intermediate data sets that JobA passes to JobB also use BatchPipes.

Changing from VIO to BatchPipes

If the intermediate data set is handled by virtual input/output (VIO), you can replace the use of VIO with BatchPipes.

Look at an example of a JCL DD statement that uses VIO:

```
//EX1  DD DSNAME=&&BDATA.TEMP,SPACE=(360,(5,30)),DCB=. . .
```

To replace VIO with a pipe named BDATA.TEMP associated with the BatchPipes subsystem named PIP2, you would need to:

- Split up the job into a writer job and a reader job.
- Supply the DCB information that BatchPipes requires on the JCL. (You might have to look in the application code for this information.)
- Change the DD statement to the following:

  ```
  //EX1  DD  DSNAME=BDATA.TEMP,SUBSYS=PIP2
  ```

- Schedule the two jobs to run simultaneously.

If VIO is handling two intermediate data sets for the jobsteps, you can use the same BatchPipes subsystem to transfer both data streams.

Using Dynamic Allocation to Specify Values

This section describes General-use Programming Interfaces and associated guidance information.

You can use dynamic allocation to make the required changes. Add the BatchPipes subsystem name text unit to your current dynamic allocation text units.

The following example allocates a pipe data set named CUSTOMER.ACCOUNTS that is to use subsystem BP01. The pipe data set is allocated to DDNAME PIPEDD, has logical record length 80, and record format of fixed-blocked. The example opens the pipe data set, writes 100 records, and closes the pipe data set.

```
         :
         :
         XC    RBAREA,RBAREA         Clear request block
         LA    R3,RBAREA             Set up base for S99RB
         USING S99RB,R3              Establish base for DSECT
         LR    R4,R3
         O     R4,=X'80000000'       Set end of list indicator
         ST    R4,RBADDR             Save address of request block
* Set up request block
         MVI   S99RBLN,LENS99RB      Set length of block
         MVI   S99VERB,S99VRBAL      Indicate allocation request
         LA    R2,TUPL               Get address of text unit pointers
         ST    R2,S99TXTPP           Set address of text unit pointers
         USING S99TUPL,R2            Establish base for text unit      x
                                     pointer list
         LA    R5,TUAREA             Text unit area address
         USING S99TUNIT,R5           Text unit mapping
* Set up dsname text unit
         ST    R5,S99TUPTR           Set address of text unit
         MVC   S99TUNIT(LENTUDSN),TUDSN  Store text unit
         LA    R5,LENTUDSN(R5)       Increment text unit ptr
* Set up ddname text unit
         LA    R2,4(R2)              Increment base for S99TUPL
         ST    R5,S99TUPTR           Set address of text unit
         MVC   S99TUNIT(LENTUDDN),TUDDN  Store text unit
         LA    R5,LENTUDDN(R5)       Increment text unit ptr
* Set up subsys text unit
         LA    R2,4(R2)              Increment base for S99TUPL
         ST    R5,S99TUPTR           Set address of text unit
         MVC   S99TUNIT(LENTUSUB),TUSUBSYS  Store text unit
         LA    R5,LENTUSUB(R5)       Increment text unit ptr
* Set up lrecl text unit
         LA    R2,4(R2)              Increment base for S99TUPL
         ST    R5,S99TUPTR           Set address of text unit
         MVC   S99TUNIT(LENTULRE),TULRECL  Store text unit
         LA    R5,LENTULRE(R5)       Increment text unit ptr
* Set up recfm text unit
         LA    R2,4(R2)              Increment base for S99TUPL
         O     R5,=X'80000000'       Indicate last pointer
         ST    R5,S99TUPTR           Set address of text unit
         MVC   S99TUNIT(LENTUBLK),TURECFM  Store text unit
* Issue the DYNALLOC macro
         LA    R1,RBADDR
         DYNALLOC                    DYNALLOC will process the request
* Open the pipe
         MVC   D_OPEN(OPENLEN),S_OPEN Set up dynamic open list
         MVC   D_OUTDCB(OUTDCBLN),S_OUTDCB Set up dynamic DCB
         OPEN (D_OUTDCB,(OUTPUT)),MF=(E,D_OPEN) Open the DCB
* Write out records
         LA   R5,100                  Writing 100 records
PUTAGAIN EQU *
         PUT  D_OUTDCB,RECORD         Write a record
         BCT  R5,PUTAGAIN
```

```
* Close the pipe
         MVC  D_CLOSE(CLOSELEN),S_CLOSE Set up dynamic close list
         CLOSE (D_OUTDCB),MF=(E,D_CLOSE) Close the pipe
         :
         :
* Define text units
TUDSN    EQU   *
         DC    AL2(DALDSNAM)                  Data set name
         DC    X'0001'
         DC    X'0011'
         DC    C'CUSTOMER.ACCOUNTS'
TUDSNE   EQU   *
TUDDN    EQU   *
         DC    AL2(DALDDNAM)                  DDNAME
         DC    X'0001'
         DC    X'0006'
         DC    C'PIPEDD'

TUDDNE   EQU   *
TUSUBSYS EQU   *
         DC    AL2(DALSSNM)                   Subsystem name
         DC    X'0001'
         DC    X'0004'
         DC    C'BP01'
TUSUBE   EQU   *
TULRECL  EQU   *
         DC    AL2(DALLRECL)                  Logical record length
         DC    X'0001'
         DC    X'0002'
         DC    X'0050'
TULREE   EQU   *

TUBLKE   EQU   *
TURECFM  EQU   *
         DC    AL2(DALRECFM)                  Record format
         DC    X'0001'
         DC    X'0001'
         DC    X'90'
TURECE   EQU   *
LENS99RB EQU   (S99RBEND-S99RB)
LENTUDSN EQU   (TUDSNE-TUDSN)
LENTUDDN EQU   (TUDDNE-TUDDN)
LENTUSUB EQU   (TUSUBE-TUSUBSYS)
LENTULRE EQU   (TULREE-TULRECL)
LENTUREC EQU   (TURECE-TURECFM)
S_OPEN   OPEN  (,),MF=L             OPEN macro list form
OPENLEN  EQU   *-S_OPEN             Length of OPEN macro
S_CLOSE  CLOSE (,),MF=L             CLOSE macro list form
CLOSELEN EQU   *-S_CLOSE            Length of CLOSE macro
S_OUTDCB DCB   DSORG=PS,MACRF=PM,DDNAME=PIPEDD
OUTDCBLN EQU   *-S_OUTDCB           Length of output DCB
RECORD   DC    CL80'DATA................................................+
               .......................DATA'
```

```
* Dynamic Area
DYNAREA  DSECT
RBADDR   DS    A
RBAREA   DS    CL50
TUPL     DS    5A
TUAREA   DS    CL256
D_OPEN   OPEN  (,),MF=L             OPEN macro in dynamic storage
D_CLOSE CLOSE (,),MF=L              CLOSE macro in dynamic storage
D_OUTDCB DCB   DSORG=PS,MACRF=PM,DDNAME=PIPEDD
         IEFZB4D0
         IEFZB4D2
         :
         :
```

See *MVS/ESA Authorized Application Development Guide* for help using dynamic allocation.

Chapter 13: Planning Recovery for Jobs Using BatchPipes

This chapter describes General-use Programming Interfaces and associated guidance information.

In planning how your jobs can recover from abends of jobs in a pipeline, you need to understand how BatchPipes handles abends.

BatchPipes's Processing for Job Abends

The BatchPipes subsystem gets control at the abend of the task that opened the piped data set. If the job associated with that task abends, BatchPipes sends I/O errors to the job's writer or reader partners, which results in the following message being recorded in each of the partner job logs:

```
ASFP365I BATCHPIPES ERROR PROPAGATED:
        JOB=jobname JOBSTEP=stepnm DD=ddnm SUBSYS=ssname
        ERRORJOB=errjobnm ERRORSYS=errsys
        ERROR-REASON: JOB errjobnm IN PIPELINE ABNORMALLY TERMINATED.
        JOB jobname WILL BE CANCELLED.
```

where JOB identifies the job that received the propagated I/O error, STEP identifies the jobstep, DD identifies the pipe, SUBSYS identifies the subsystem name, and ERRORJOB identifies the jobname and sysname of the job that abended. For information about how the partners react to the I/O error, see "Alerting Pipe Partners about a Job's Abend."

Note that BatchPipes monitors the task associated with the job that performs the open, but not necessarily the task associated with the job that requests I/O to the pipe. If one task opens the data set and a different task requests the I/O and then abends, BatchPipes will **not** detect the abend of that second task and will not alert other jobs connected to the pipe. The outcome is no different from jobs in a non-BatchPipes jobstream.

Alerting Pipe Partners about a Job's Abend: When a job using a pipe abends, **all other jobs with connections to the pipe receive an I/O error**, which might cause those jobs to abend, depending on how the EROPT DCB parameter, or ERRPROP Subsys keyword, is specified. The I/O error usually causes the jobs connected to the pipe to be CANCELled (ABEND x'222'), in which case any data in the pipe and in buffers is lost and the pipe connections close. Most likely you would re-run those jobs.

The default setting for EROPT DCB parameter is 'ABE' which results in the following behavior:

Figure 44 shows two cases in which jobs named JOBF abend. Other jobs connected to that pipe receive I/O errors and are CANCELled (via the MVS CANCEL jobname,A=asid console command). In Case 1, JobF and Job A are the writer/reader partners. JobF abends causing JobA to receive an I/O error and be CANCELled. In Case 2, JobF abends, JobB and JobA receive I/O errors and are CANCELled.

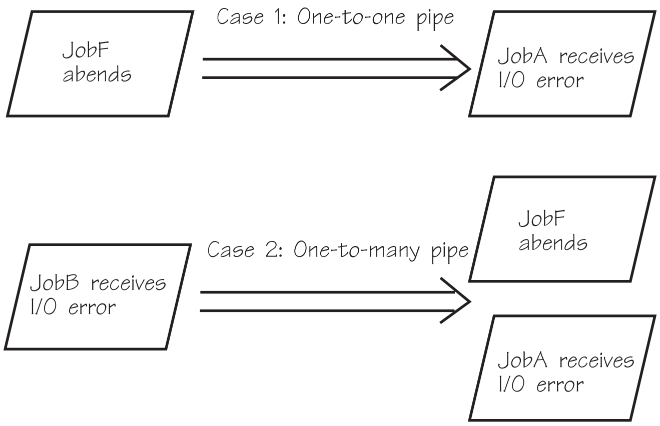


Figure 44. Abend Processing Example

Tolerating a Partner's I/O Error: Some jobs, such as those that take samples of data, can tolerate lost records. In these cases, you can enable some jobs to continue to process records; for example, a reader can process records that remain in a pipe. Any records in the partner job's buffer are lost.

To allow processing to continue, use the EROPT=SKP or EROPT=ACC parameter on the DCB statement, or 'ERRPROP=CONT' or 'ERRPROP=DUMMY' on the SUBSYS option, that defines the data set (the pipe). The default for EROPT is ABE, causing an abend when an I/O error is received. (The example earlier in this section assumes that JobA used the default value on EROPT.) If EROPT=SKP or ACC are specified and the I/O error is received by a job using a pipe, one of the following messages appears in the job log:

```
ASFP375I BATCHPIPES I/O ERROR PROPAGATED:  EROPT=SKP
        JOBSTEP=stepnm DD=ddnm SUBSYS=ssnm PARTNER=(jobid,jobname,jobstep)
                            or
ASFP376I BATCHPIPES I/O ERROR PROPAGATED:  EROPT=ACC...
```

where JOBSTEP identifies the jobstep that received the propagated I/O error, DD identifies the pipe, SUBSYS identifies the subsystem name, and PARTNER identifies the jobid, jobname, and stepname of the job that abended.

In Figure 44, if EROPT=ACC is coded for JobA and JobB, and JobF abends,

- In Case 1 with the one-to-one pipe, JobF's connection closes. Any records in JobF's buffers at the time of the abend are lost. JobA receives an I/O error and continues to read any remaining records in the pipe. When the pipe is empty, JobA receives an end-of-file indication and completes its processing.
- In Case 2 with the one-to-many pipe, JobF's connection closes. JobA and JobB receive I/O errors but continue processing. When JobB finishes running and the pipe is empty, JobA receives an end-of-file indication and completes its processing. Any records in JobF's buffers at the time of the abend are lost.

In Figure 44, if EROPT=ACC is coded for JobF and JobB, and JobA abends,

- In Case 1 with the one-to-one pipe, JobA's connection closes. Any records in JobA's buffers at the time of the abend are lost. JobF receives an I/O error and continues to write records to the pipe until either JobF completes or the pipe becomes full.

  Note: Caution should be used when coding EROPT=ACC on a Writer job, as in this case, because of this possible loss of data/hung job scenario.

- In Case 2 with the one-to-many pipe, JobA's connection closes. JobF and JobB receive I/O errors but continue processing. When JobB finishes running and the pipe is empty, JobF receives an end-of-file indication and completes its processing. Any records in JobA's buffers at the time of the abend are lost.

In Figure 44 on page 136, if EROPT=SKP is coded for JobA and JobB, and JobF abends,

- In Case 1 with the one-to-one pipe, JobF's connection closes. Any records in JobF's buffers at the time of the abend are lost. JobA receives an I/O error and continues to read any remaining records in the pipe. When the pipe is empty, JobA receives an end-of-file indication and completes its processing.
- In Case 2 with the one-to-many pipe, JobF's connection closes. JobA and JobB receive I/O errors but continue processing. When JobB finishes running and the pipe is empty, JobA receives an end-of-file indication and completes its processing. Any records in JobF's buffers at the time of the abend are lost.

In Figure 44 on page 136, if EROPT=SKP is coded for JobF and JobB, and JobA abends,

- In Case 1 with the one-to-one pipe, JobA's connection closes. Any records in JobA's buffers at the time of the abend are lost. JobF receives an I/O error and stop writing data to the pipe but would continue processing and writing data to its DD. In this case you would probably want to code a 'hardening' fitting here so that the writer job (JobF) can run to completion and, when Job A fails, restart it with the hardened copy of the data.
- In Case 2 with the one-to-many pipe, JobA's connection closes. JobF and JobB receive I/O errors but continue processing. When JobB finishes running and the pipe is empty, JobF receives an end-of-file indication and completes its processing. Any records in JobA's buffers at the time of the abend are lost.

Strategy for Recovery

When you consider how to recover from an abend, keep in mind the following facts:

- When one job in a pipeline fails, all jobs stop running (unless you have coded EROPT=ACC or SKP)
- The elapsed time improvements for jobs using BatchPipes might be so significant that you can rerun the jobs that abend and still save elapsed time.
- BatchPipes tolerates checkpoint requests, but fails restart requests.
- The use of BatchPipes removes two common causes for job failures: those failures associated with:
  - Physical magnetic media errors
  - Out-of-space abends.

To plan for recovery, examine your batch jobstream for logical breakpoints, those points where you can store all output records on physical media. The breakpoint is called a **recovery boundary**. Then, if an abend occurs with “downstream” jobs, you can rerun the jobs from that recovery boundary. See “Writing a Copy of Data to Tape or DASD” on page 148 for an example of a filter job that allows data to be written to DASD or tape and also to a reader job, a process known as **hardening data**.

Because most sort utilities produce multiple copies of the output file, sorts can easily serve as recovery boundaries. For example, a sort could read data from a pipe and write data to both a pipe and an external device. If you need to restart the jobs, starting from the recovery boundary, you can override the JCL of the job reading the sort output. The override statement includes the physical data set name for the pipe name.

Chapter 14: Security Considerations

This chapter describes ways that you can prevent unauthorized persons or jobs from using the pipes. Reasons why you would want the protection include the following:

- If you charge for the use of pipes, you do not want anyone except authorized users to run jobs using BatchPipes
- To control the use of the resources that pipes require, you might want to limit the use of pipes.
- It is possible for a user to, inadvertently or deliberately, submit a job that would join a pipeline that is not intended for its use. The result is that the input to the pipe includes unwanted records or the output has missing records.

  For example, consider JobA that passes data through a pipe to JobB. Another job could join JobA as a writer and add unwanted input to JobB. One logical occurrence of this scenario could be when two versions of a job exist: one for production and one for testing. If the purpose of the job is to credit money to a bank account, the additional records or lost records could cause big problems.

To provide the necessary protection:

- Use a security product such as RACF to protect the use of a BatchPipes subsystem. See “Controlling the Use of BatchPipes” on page 34.
- To protect a pipe, use the same generic protection that you use in the non-BatchPipes environment to protect a data set.

Chapter 15: Changing Applications to use BatchPipes

This chapter describes changes you might make to the applications themselves. Such changes include:

- Making the data movement more efficient, such as removing data contention or cloning jobs
- Changing the flow of data, such as using filter jobs
- Redesigning an entire application to maximize the use of BatchPipes.

Reducing Contention for Data

Most of the time, jobs in pipelines also use non-BatchPipes data that resides on physical devices. Sometimes two or more jobs in a pipeline use the same non-BatchPipes data; sometimes they share multiple data sets that reside on one device. When this kind of sharing exists, contention for use of the data can slow down the pipeline. Cloned jobs that share the same data base are an example of this situation.

This section describes a scenario where contention for data on DASD slows down the pipeline; it suggests three ways you can change your application to reduce the contention.

Let's look at an example of a company that has a billing application that consists of three jobs that run one after the other:

- JOBA, which reads all the day's new orders and verifies that the order is correct
- JOBB, which reads pricing information on a data base and creates a price for each order
- JOBC, which creates the bills.

To shorten the elapsed time of the application, the company decides to use BatchPipes so that JOBA, JOBB, and JOBC are a pipeline as illustrated in Figure 45:

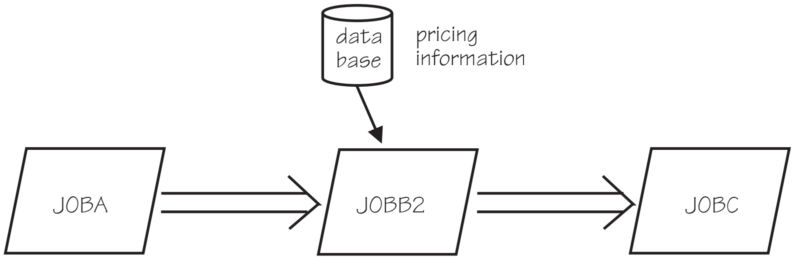


Figure 45. A Three-Job Pipeline

With the jobs using BatchPipes, the company discovers that JOBB is slow relative to its two partners. For this reason, the application developers clone JOBB so that three copies of the job run, as illustrated in Figure 46 on page 142. The pipeline now consists of JOBA, three cloned JOBBs reading from the same pipe, and JOBC.

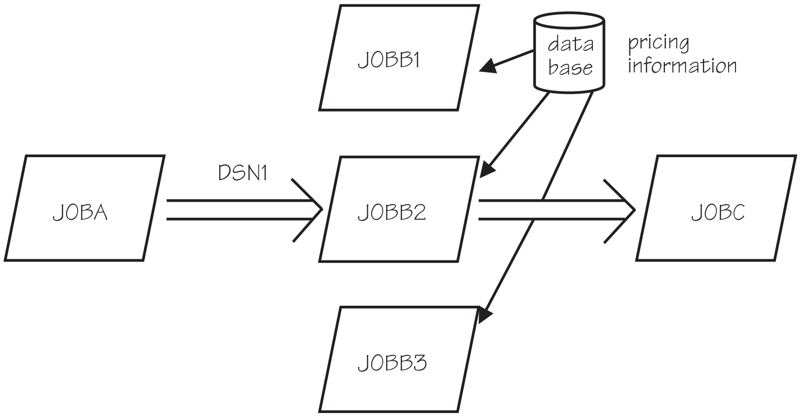


Figure 46. Three Cloned JOBBs using a Data Base

With the three jobs reading the pricing information from the same data base, JOBB1, JOBB2, and JOBB3 are all in the pipeline, contending for the data. Three ways to reduce contention for the data base are:

- Scenario 1: Sharing a buffer pool
- Scenario 2: Partitioning the shared data base
- Scenario 3: Using Hiperbatch.

Scenario 1: Sharing a Buffer Pool: One way to reduce contention for the data base is to use a shared buffer pool. With JOBB1, JOBB2, and JOBB3 having a shared buffer pool, the cloned jobs can read data from the same set of buffers. This sharing of buffers results in fewer physical I/O operations per job, reducing contention for the pricing information.

If you run DB2 jobs, for example, the cloned jobs can share system-wide buffer-pools. These pools give optimal support to direct (random) processing.

Scenario 2: Partitioning the Data: The second way to reduce contention is to partition the data so that JOBB1, JOBB2, and JOBB3 each have their own source of pricing information, according to geographical area. Each reads from a separate pipe and writes through pipes to cloned JOBCs. Each JOBC clone creates a bill for a specific geography. This solution, described in Figure 47, requires logic changes to the application; JOBA must route records to the cloned JOBBs according to geography.

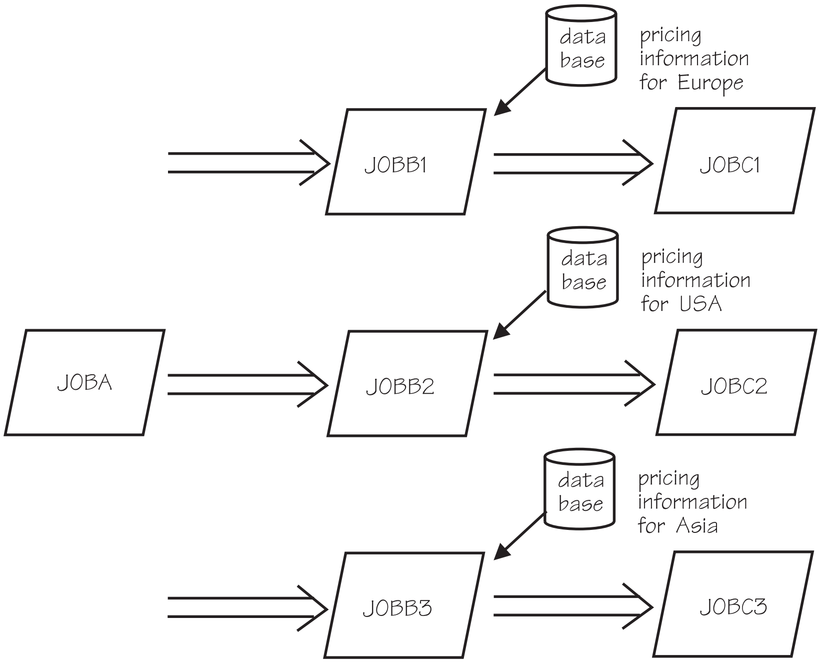

data base
pricing information for Europe
JOBB1
JOBC1
data base
pricing information for USA
JOBA
JOBB2
JOBC2
data base
pricing information for Asia
JOBB3
JOBC3

Figure 47. Partitioning the Shared Data Base

Scenario 3: Using Hiperbatch:

With hiperbatch, cloned jobs can share data sets concurrently using QSAM or VSAM Non-Shared Resources (NSR). In this scenario, the pricing information would either be sequentially accessed (which is unlikely) or have been loaded into a hiperbatch RETAINed object by a previous job or jobstep.

Cloning a Writer or Reader

This section provides guidance needed for you to make multiple copies of jobs. You would “clone” jobs for two reasons: to improve the balance of a pipe and to concentrate more processors on your batch jobstream

To clone the job, make exact copies of the JCL for the job, each copy having a unique job name. Then, make sure enough initiators exist for the cloned jobs. Finally, schedule the jobs to run at the same time.

Before you actually make the copies of the jobs, consider how the data will flow in the pipeline. The order that records flow **from** multiple writers and **to** multiple readers is unpredictable. See “Notes on Data Transfer for Multiple Writers and Readers” on page 113 for a summary of data flow when a pipe has multiple writers or readers.

Figure 48 on page 144 illustrates the case where the writer writes approximately twice as fast as the reader reads. Note that the order of the output records from Job2 and Job2A will not be the same as the output from Job1. If order is important, include a merge program later in the pipeline.

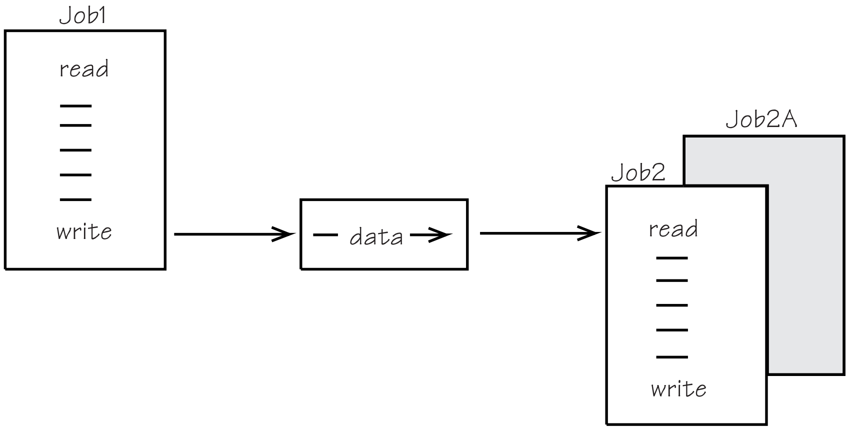


Figure 48. A Writer Job and Two Reader Jobs Using a Pipe

After you identify that a job is slowing down the progress of records through a pipe, how many copies should you make?

- Try estimating the relative speed of jobs as they ran before BatchPipes. Obtain a rough estimate of relative speeds of writer and reader by comparing the EXCP rates for data to be piped. If the readers reads 200,000 EXCPs to the data set over 20 minutes (that is 10,000 EXCP per minute) and the writer writes 200,000 EXCPs over 10 minutes (20,000 EXCPs per minute), you can reasonably expect two readers to balance the one writer.
- Another way, both simple and effective, is to start with a small number of cloned jobs, say three or four, and then use the BatchPipes STATUS command to see how busy the jobs are. If one or more of the “cloned” jobs are considerably less busy than others, the next time you run the jobs, eliminate the jobs that are less busy. Remember that each cloned job uses some system resources.

If you have writer/reader jobs that are suitably balanced, you might consider cloning them both. If they are running when processor usage is low, you can make better use of the processor **and** get significant elapsed time improvements for the jobs. Clone the partners so that the pipeline consists of *n* writers and *n* readers. Most likely you would use this approach during less busy hours.

When duplicate writers and readers use the pipe, data-partitioning becomes an important consideration. See “Reducing Contention for Data” on page 141.

Some Considerations when Cloning Jobs

When you clone writer and reader jobs, some pitfalls to be aware of are:

- If the reader job counts records and processes a specific number of records before it closes the input file, the job cannot be cloned without rewriting the code. Change the code so that the end-of-file indication from the writer triggers the closing of the input file by the reader jobs.
- Clone reader jobs only when the job has record independence; that is, when it processes each record individually and doesn't need to see a set of records within one copy of the job.

Rules for coding the JCL parameters are the same whether multiple jobs or single jobs. BLKSIZE values can be different for readers and writers as long as the BLKSIZE value (or values) for reader (or readers) is greater than or equal to the BLKSIZE value (or values) for writer (or writers). Again, **IBM recommends** that you omit the BLKSIZE parameter.

Multiple Distinct Readers or Writers Using One Pipe

It is possible for two or more distinctly different reader jobs to read all records from a writer job. This section offers two possibilities for multiple distinct reader jobs to process the same data.

It is possible for you to change the applications so that both readers can read all records from the single writer. The simplest way to describe the topic is through an example in which two different applications APPLB and APPLC want to read every record in data set DSN1 which application APPLA writes. **Every** record that APPLA writes becomes input to **both** APPLB and APPLC. The first scenario requires an additional program, the second requires additional code in APPLA.

First scenario: In the first scenario, illustrated in Figure 49, APPLA writes to the pipe DSN1. A program called COPY2 reads from the pipe and writes each record twice, once to pipe named DSN1B and once to pipe named DSN1C. APPLB and APPLC then read from DSN1B and DSN1C.

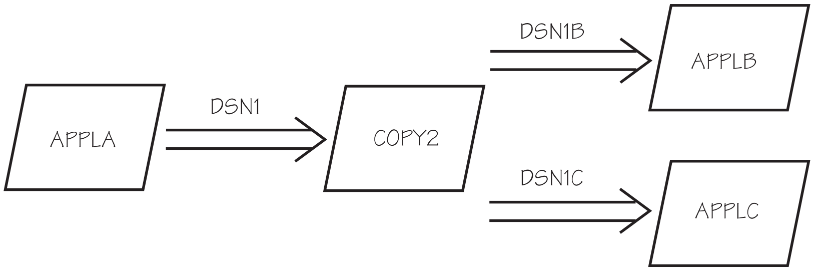


Figure 49. Two Different Readers Processing the Same Data: First Scenario

The program named COPY2 is a filter job — one that uses a pipe to change the way data flows in the pipeline and uses additional processor resources. Design COPY2 so it can handle a flexible number of readers.

Second scenario: In the second scenario, illustrated in Figure 50, you define a data set description for DSN2 within APPLA, and have APPLA issue two identical write requests for each record, one for DSN1 and one for DSN2.

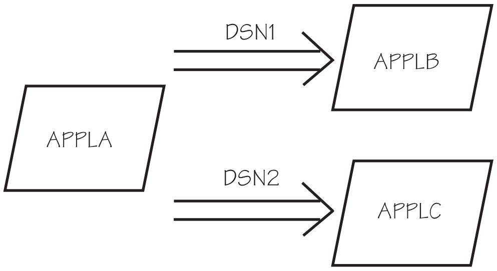


Figure 50. Two Different Readers Processing the Same Data, Second Scenario

Comparison of first and second scenarios: The advantage of the second scenario over the first is that far fewer write and read requests occur and, therefore, processor overhead is less. The disadvantage is that the second scenario requires changes to application code; if you want to add a third reader, the application code will need to change again. Changes to COPY2 to write to an additional job are insignificant.

Replacing Concatenated Data Sets with Pipes

Concatenated data sets are read in sequential order. The first data set is opened, read, and closed; likewise for the second, and so forth. If you keep this fact in mind along with the advanced data set concepts described “How BatchPipes Interprets OPEN and CLOSE Macros” on page 109, you can change some of your existing concatenated data sets to use BatchPipes.

One good use of BatchPipes is when a number of jobs create data sets that then become input to a single job. Consider the following situation where JOBA creates the data set named DSN1 and JOBB creates the data set named DSN2, both on tape drives. JOBC reads concatenated data sets DSN1 and DSN2 in the following way. When JOBA closes DSN1, JOBC reads DSN1; when JOBB closes DSN2, JOBC reads DSN2. JOBA and JOBB can run in parallel.

Figure 51 shows the three jobs' use of the data sets and the JCL statements:

```
JOBA creates DSN1
      //SYSUT2 DD DSN=DSN1,DISP=(NEW,CATLG),UNIT=TAPE,LRECL=123,RECFM=FB

JOBB creates DSN2
      //SYSUT2 DD DSN=DSN2,DISP=(NEW,CATLG),UNIT=TAPE,LRECL=123,RECFM=FB

JOBC reads concatenated data sets DSN1 and DSN2
      //SYSUT1 DD DSN=DSN1,DISP=(old,keep)
      //       DD DSN=DSN2,DISP=(old,keep)

The timeline looks like this:

 <----JOBA creates DSN1--><---JOBC reads DSN1---
 <----JOBB creates DSN2-->                      ---JOBC reads DSN2->
 ____________________________________________________________________
 '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '  '
```

Figure 51. Example of Concatenated Data Sets

You can improve the elapsed time by using one or more pipes instead of one or both data sets on physical media.

First scenario: In the first scenario, illustrated in Figure 52 on page 147, DSN1 and DSN2 remain concatenated. JOBA writes to pipe DSN1; running in parallel with JOBA, JOBC reads from DSN1. When JOBC closes the piped data set DSN1, JOBB can start writing to pipe DSN2; running in parallel with JOBB, JOBC reads from DSN2. The first scenario eliminates tape mounts.

```
JOBA writes to pipe DSN1
     //SYSUT2 DD DSN=DSN1,SUBSYS=BP01,LRECL=123,RECFM=FB

JOBB writes to pipe DSN2
     //SYSUT2 DD DSN=DSN2,SUBSYS=BP01,LRECL=123,RECFM=FB

JOBC reads from pipe DSN1 first, then reads from pipe DSN2
     //SYSUT1 DD DSN=DSN1,SUBSYS=BP01,LRECL=123,RECFM=FB
     //       DD DSN=DSN2,SUBSYS=BP01,LRECL=123,RECFM=FB

The timeline looks like this:

 < . . JOBB waits . . . -----JOBB writes DSN2--->
 <---JOBA writes DSN1-->
 <---JOBC--reads DSN1--><----JOBC reads DSN2---->
```

Figure 52. Two Concatenated Data Sets Defined as Pipes

Second scenario: The second scenario introduces more parallel processing and eliminates the concatenation of the two data sets. In the second scenario, illustrated in Figure 53, JOBA and JOBB both write to the pipe DSN3, and JOBC reads from the pipe.

```
JOBA writes to pipe DSN3
     //SYSUT2 DD DSN=DSN3,SUBSYS=BP01,LRECL=123,RECFM=FB

JOBB writes to pipe DSN3
     //SYSUT2 DD DSN=DSN3,SUBSYS=BP01,LRECL=123,RECFM=FB

JOBC reads from pipe DSN3
     //SYSUT1 DD DSN=DSN3,SUBSYS=BP01,LRECL=123,RECFM=FB

The timeline looks like this:

 <----JOBA writes to DSN3-->
 <----JOBB writes to DSN3-->
 <----JOBC reads from DSN3->
```

Figure 53. Concatenated Data Sets Replaced with One Pipe

This solution requires that JOBC be able to process the records from the pipe DSN3 in no particular order. A sort or merge, run just before or after JOBC, could reorder the records. Sometimes record order is not actually important to the application.

Writing a Copy of Data to Tape or DASD

In “Multiple Distinct Readers or Writers Using One Pipe,” you see two possibilities for multiple distinct jobs to process the same data. In a similar way, the data that was read by APPLB could be written to a physical device, an example of hardening data. The filter job COPY2 in Figure 49 on page 145 would write each record to tape or DASD and each record to APPLC. Or, you would change APPLA to write one copy of each record to DASD or tape and one copy of each record to APPLC, as in Figure 50 on page 145.

When you harden data, you use the I/O subsystem; however, you still gain the benefits of concurrent processing of the jobs.

Redesigning Applications to Use BatchPipes

Before you begin any redesigning efforts, weigh the cost of the programming effort against the improvement in elapsed time. Choose jobs that have the characteristics that are most likely to gain the benefits from BatchPipes:

- The read and write actions in the jobs are I/O intensive
- The jobs are long-running
- The jobs are in the critical path
- The jobs use tape as storage media.

Some batch jobs have been around for a long time; they were written to run on computers that were a fraction as powerful as the present-day computers and had far less processor storage available. Few jobs have been designed to fully exploit the capabilities of modern computers. In redesigning jobs for BatchPipes, you can take advantage of the processor's ability to multiprocess. For example, most jobs consist of a single task. Consider splitting up the job into three jobs, each running under a different task, each using pipes. You can, for example, split up a single long-running job into:

- A “front end job” that reads from tape and writes to a pipe
- A “driver job” that reads the data from the pipe, processes it, and writes to a pipe
- A “back end job” that reads the data from the pipe and writes to tape.

This scenario involves many additional read and write requests. One question you might ask as you look at this scenario is: how much does the parallel processing that BatchPipes offers compensate for the additional requests.

This is a bold move, but, carefully done, it could bring you very large performance gains.

Chapter 16: Questions and Answers for Application Developers

This chapter lists questions that application developers have asked and the answers they received.

- **Question**: Does BatchPipes support the DISP=CATLG parameter on the DD statement that defines a pipe?

  Answer: No. BatchPipes ignores all uses of DISP='.

- **Question**: Which writer's or reader's BUFNO specification defines the number of blocks in a pipe?

  Answer: The first job whose piped data set is opened.

- **Question**: Do all COBOL write and read instructions work with BatchPipes?

  Answer: Yes, any read and write statement that represents a sequential QSAM or BSAM request.

- **Question**: How does a reader job find out that a writer job abends?

  Answer: When one job abends, all other jobs with connections to the pipe receive an I/O error.

- **Question**: If there are multiple readers and one writer, will all readers receive the end-of-file indication and the I/O error message?

  Answer: Yes.

- **Question**: If a job without BatchPipes is writing its data to a member of a partitioned data set that becomes input to the next job in the jobstream, can a pipe replace that partitioned data set member?

  Answer: While BatchPipes supports sequential data sets only, if the data set is a temporary repository, the jobs can use BatchPipes. You will need to change the DSN parameter on the statement that describes the data set.

- **Question**: Does BatchPipes work with checkpoint/restart?

  Answer: BatchPipes tolerates checkpoint requests, but does not perform any checkpoint functions. BatchPipes does not support restart requests.

- **Question**: Does BatchPipes allow multiple opens and closes of a piped data set?

  Answer: As long as there is one open connection to a piped data set, the data set can be opened and closed multiple times. However, open requests will not complete unless there is at least a writer/reader pair, both of which have issued an open request to the pipe unless OPENNOW is specified on the SUSBSYS parameter.

- **Question**: If one program references a pipe (for example, DSN=MY.OLD.DASD.FILE,SUBSYS=BP on the DD statement) and another program references dataset named MY.OLD.DASD.FILE), will the references conflict?

  Answer: They will not conflict.

- **Question**: Can I define a pipe through dynamic allocation?

  Answer: Yes, by adding the text unit for SUBSYS to your existing dynamic allocation invocation.

- **Question**: Can one job in a pipeline check the condition code of another job in the pipeline?

  Answer: No, condition codes are not “passed” from job to job within a pipeline. Because the jobs run in parallel, one job in the pipeline cannot rely on a condition code from another job to determine its processing.

- **Question**: Can a job in a pipeline use an auditing routine?

Answer: You need to understand how the results of the audit are used. No problem exists if the results are used by a job that runs after the pipeline completes. Be wary, however, if the processing of a later job in the pipeline is dependant on the results of the audit of an earlier job in the pipeline.

- **Question**: Can a job in a pipeline use an auditing or balancing routine?

  Answer: You need to understand how the results of these routines are used. No problem exists if the results are used by a job that runs after the pipeline completes. Be wary, however, if the processing of a later job in the pipeline depends on the results of the audit of an earlier job in the pipeline. For example, a routine that counts records must finish running before the total count is complete. If a job at the next pipeline stage reads the count field at the beginning of its run, the count is most likely not complete.

Section V: Appendixes

Appendix A: SMF Record Descriptions

This chapter describes General-use Programming Interfaces and associated guidance information.

Record type 91 (X'5B') is written at START and STOP of the BatchPipes subsystem, at the subsystem recording interval, at OPEN and CLOSE of a pipe, and at pipe creation and deletion. The records contain information about the BatchPipes subsystem, and activity such as counts for pipes, pipe allocations, pipe OPENs, and pipe CLOSEs.

Record Environment

The following conditions exist for the generation of this record:

Mode:	Task
Length of record:	Variable
Storage residency:	Above 16 megabytes
Macro:	SMFEWTM, branch entry (record exit: IEFU84)
Locks:	No locks held
Address space:	The BatchPipes subsystem
SUBSYS:	The BatchPipes subsystem

Subtypes of Type 91 Records

The eight subtypes for Type 91 records are:

- **Subtype 1, written at subsystem start**, described on page 155.
- **Subtype 2, subsystem activity written at end of the subsystem recording interval**, is described on page 156.
- **Subtype 3, written at subsystem termination**, is described on page 157.
- **Subtype 11, written at pipe OPEN connection**, described on page 158.
- **Subtype 12, pipe activity written at the end of the subsystem recording interval**, described on on page 159.
- **Subtype 13, written at pipe CLOSE connection**, described on page 160.
- **Subtype 14, written at pipe creation**, described on on page 161.
- **Subtype 15, written at pipe deletion**, described on on page 162.

Sections of Type 91 Records

Each Type 91 record begins with a header section, a self-defining section, and up to eight different sections. The self-defining section consists of a set of triplet fields that you should use to locate the other sections on the record:

- Offset from the beginning of the record (field SMF91*xx*O)
- Length of record (field SMF91*xx*L)
- Number of sections on the record (field SMF91*xx*N)

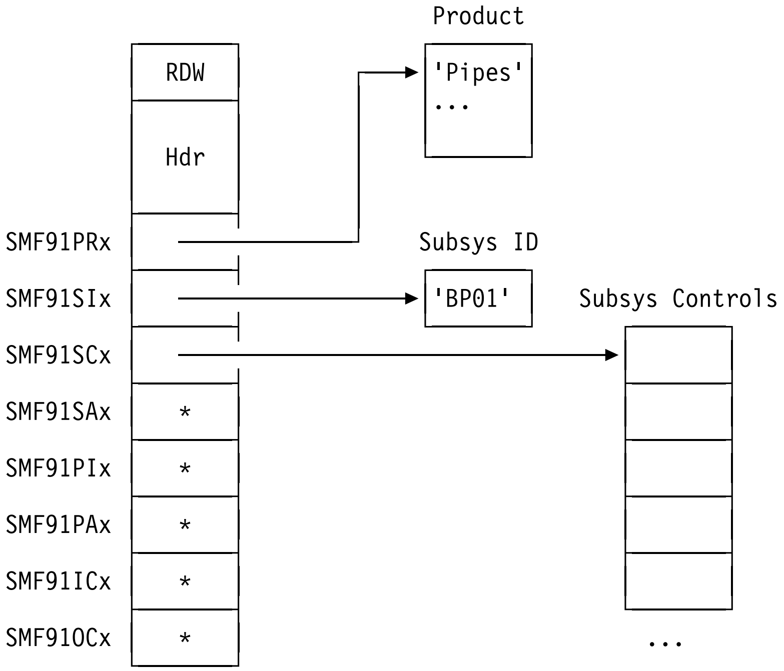
The eight sections are:

- The **product section** (triplet fields named SMF91PR*x*) provides some general identification information about BatchPipes and the system at the time the record was generated. The product section appears on all records.
- The **subsystem identification section** (with triplet fields named SMF91SI*x*) provides general identification and parameter values for the specified BatchPipes subsystem. The subsystem identification section appears on all records.
- The **subsystem controls section** (with triplet fields named SMF91SC*x*) provides general parameter values for the specified BatchPipes subsystem. It appears only on subsystem-related records (subtypes 1, 2, and 3).
- The **subsystem activity section** (with triplet fields named SMF91SA*x*) provides usage statistics for the specified BatchPipes subsystem. It appears only on subsystem-related records (subtypes 2 and 3).
- The **pipe identification section** (with triplet fields named SMF91PI*x*) provides the identification for the pipe that this record represents. It appears only on pipe-related records (subtypes 11, 12, 13, 14, and 15).
- The **pipe activity section** (with triplet fields named SMF91PA*x*) provides the summary usage statistics for the pipe that transfers this record. The section appears on pipe-related records (subtypes 11, 12, 13, and 15) only if there is data for SMF to report. The section does not appear on the pipe-creation record (subtype 14).
- The **pipe writer connection section** (with triplet fields named SMF91IC*x*) provides the specific connection information for writer to the pipe. The section appears on pipe-related records (subtypes 11, 12, 13 and 15) only if there is data for SMF to report. The section does not appear on the pipe-creation record (subtype 14).
- The **pipe reader connection section** (with triplet fields named SMF91OC*x*) provides the specific connection information for each user of this pipe. It appears on pipe-related records (subtypes 11, 12, 13 and 15) only if there is data for SMF to report; it does not appear on the pipe creation record (subtype 14).

Diagrams of the Subtypes

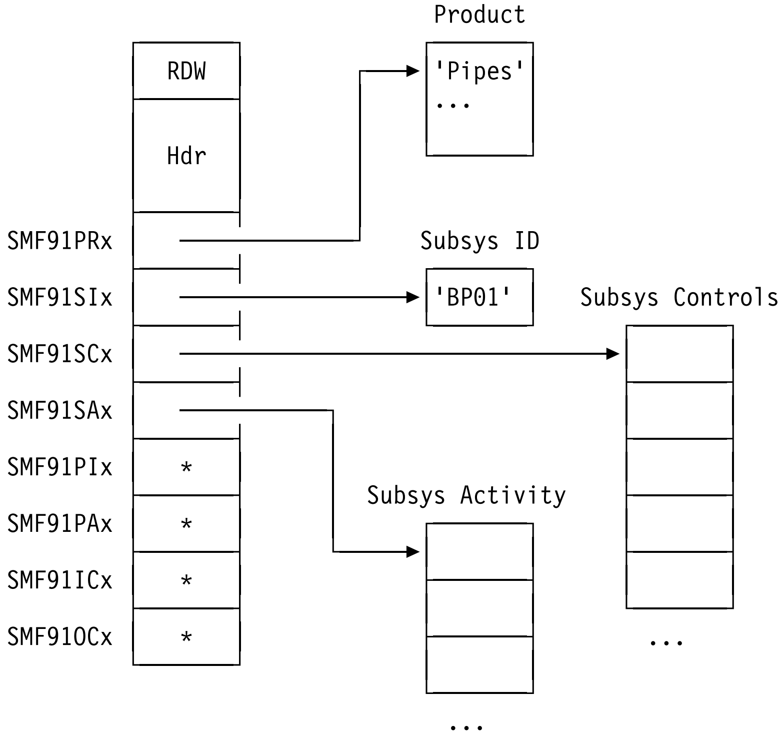
The following diagrams show the format of each Type 91 record. They all start with the header (and self-defining triplet) and are followed by the sections mentioned in “Sections of Type 91 Records” on page 154.

Subtype 1 - Subsystem Start: This record is written when the requested BatchPipes subsystem is started. It contains data pertaining to the initialization of the subsystem.



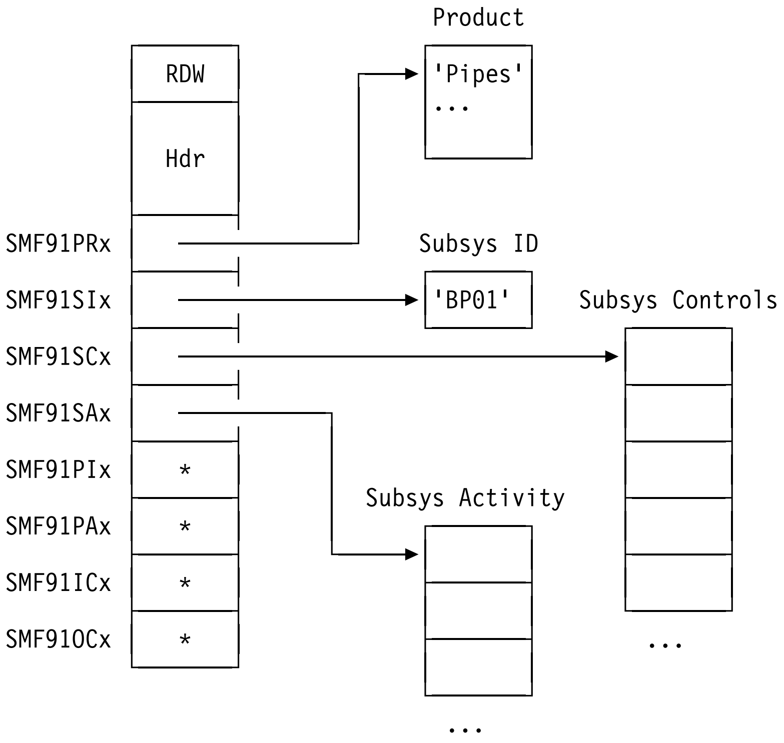
* indicates that Subtype 1 does not include this section.

Subtype 2 - Subsystem Recording Interval: This record is written at the end of the subsystem recording interval. It contains data that indicates changes in the numbers of pipes, allocations, connections, opens, and closes over the recording interval. It is written only if the subsystem recording interval is initialized in SMFPRMxx parmlib member.



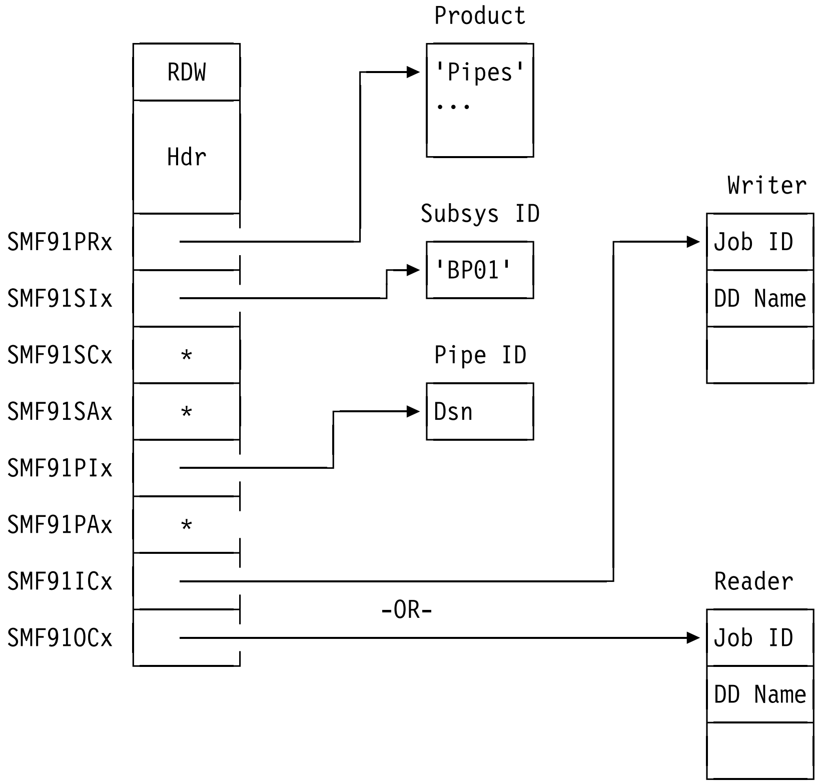
* indicates that Subtype 2 does not include this section.

Subtype 3 - Subsystem Termination: This record is written at subsystem termination. It contains data that indicates the status of the subsystem at subsystem termination. Depending on how the subsystem terminated, this record may not always be available. If a subtype 3 record exists, a subtype 2 record is written to indicate the activity during the interval between the last subtype 2 recording and subsystem ending.



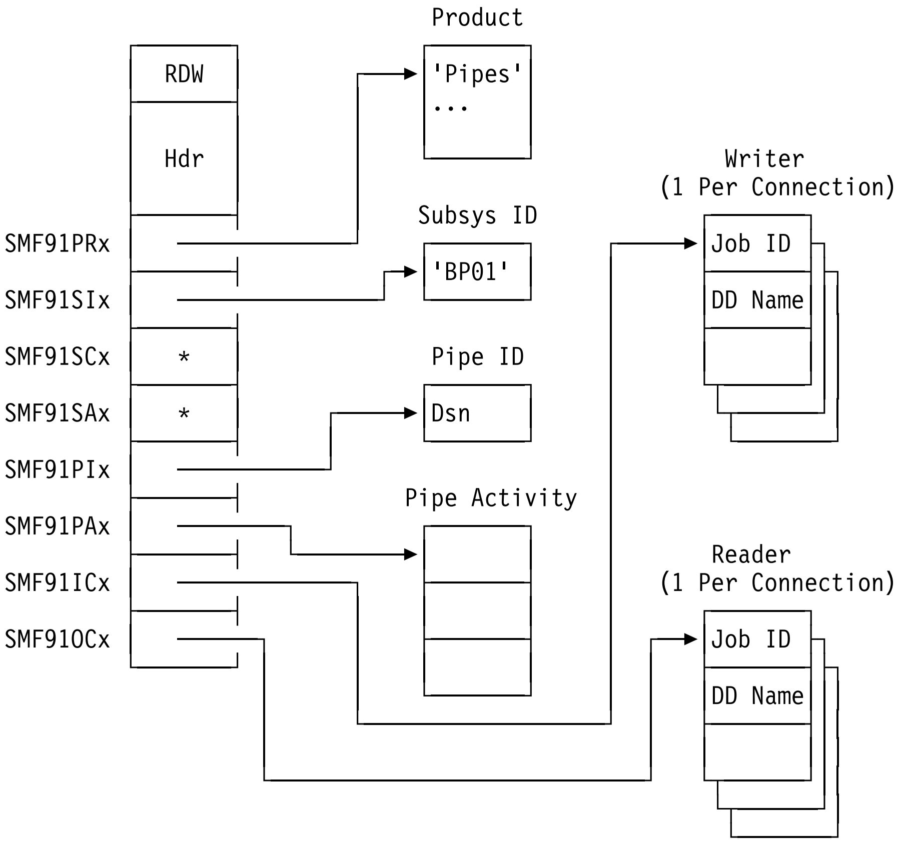
* indicates that Subtype 3 does not include this section.

Subtype 11 - Pipe OPEN Connection: This record is written whenever a user connects to a Pipe. It contains open connection date, time, DCB characteristics and user identification information.



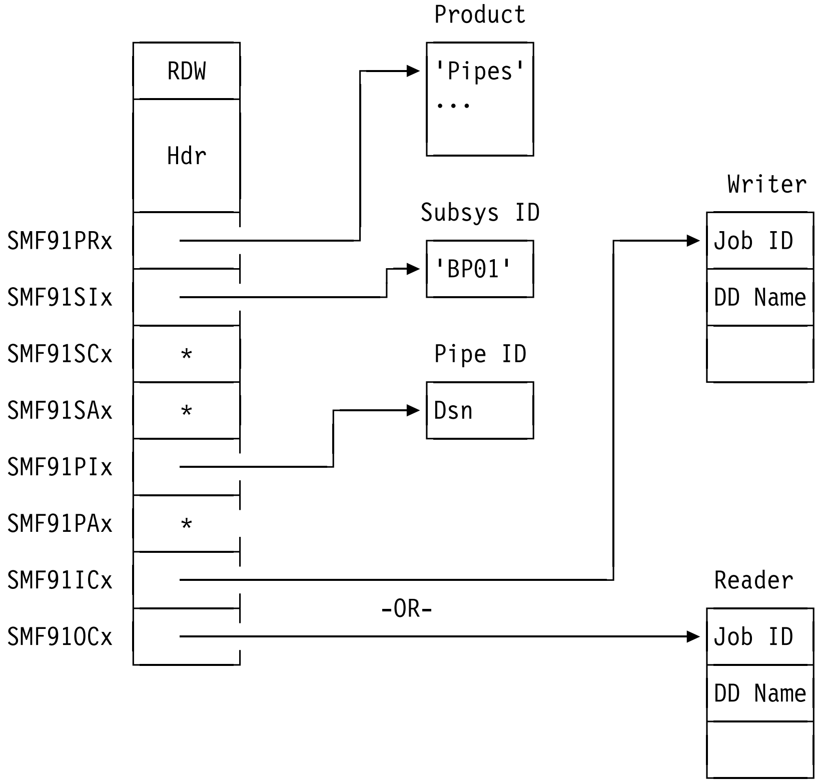
* indicates that Subtype 11 does not include this section.

Subtype 12 - Pipe Interval Record: This record is written at the end of the subsystem recording interval for the subsystem AND prior to each pipe CLOSE and DELETE record. It contains the pipe information and a summary of the activity on the pipe since the beginning of the interval. It also contains job information for ALL the connections to the pipe during the interval. It is written only if the subsystem recording interval is initialized in SMFPRMxx parmlib member.



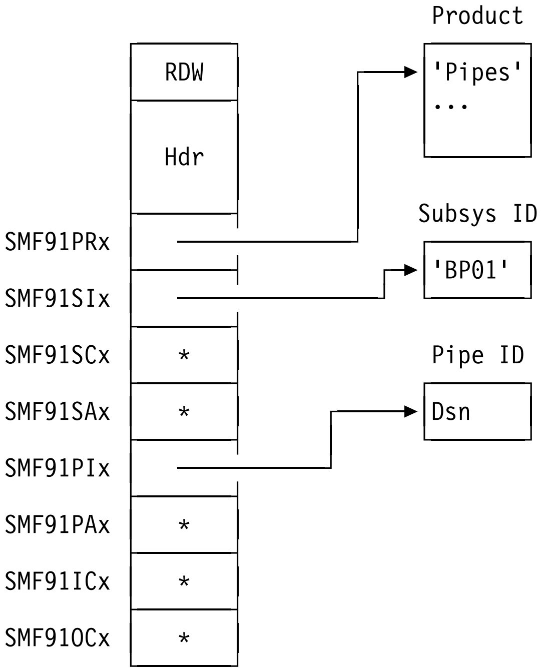
* indicates that Subtype 12 does not include this section.

Subtype 13 - Pipe Connection Close: This record is written when the user closes the connection to the pipe. It contains the pipe and job information for the connection to the pipe as well as a summary of activity for this connection to the pipe. Information includes OPEN connection date, time, DCB characteristics, CLOSE connection date and time, and user identification.



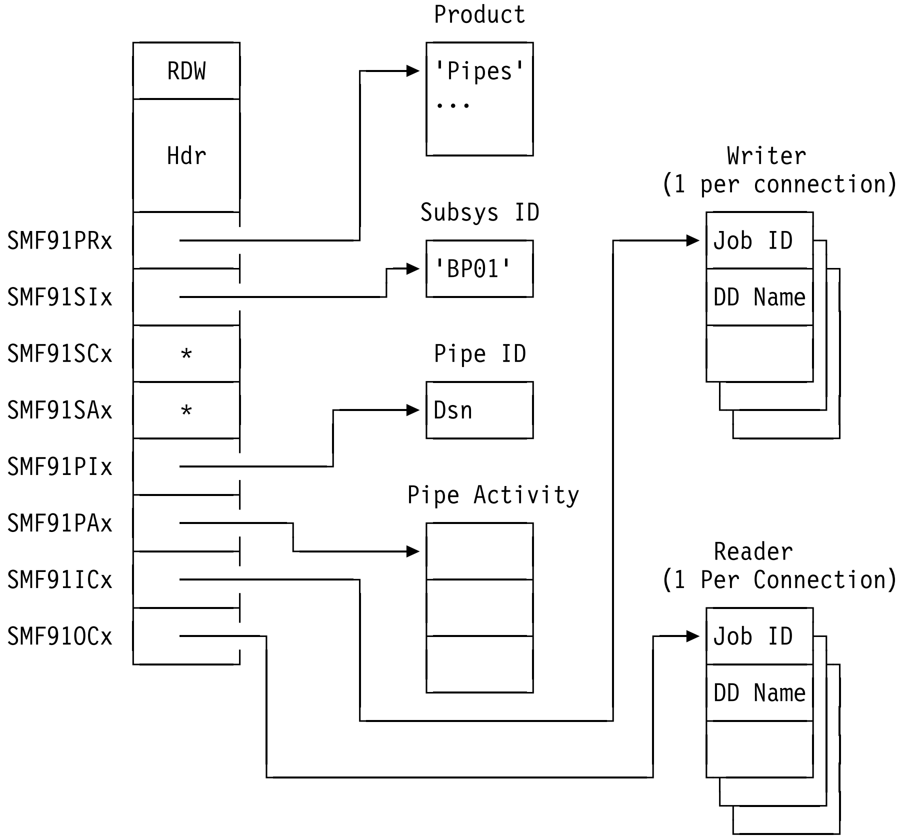
* indicates that Subtype 13 does not include this section.

Subtype 14 - Pipe Create: This record is written when the FIRST user connects to a pipe. It contains general identification information about the pipe.



* indicates that Subtype 14 does not include this section.

Subtype 15 - Pipe Delete: This record is written when the last connection to a pipe is deleted. It contains the pipe information and a summary of the activity on the pipe since it was created. It also contains job information for ALL the connections to the pipe and a summary of the activity for EACH connection.



* indicates that Subtype 14 does not include this section.

Type 91 Record Mapping

The following sections contain the full mapping of the SMF Type 91 record for BatchPipes.

Record Header

The record header appears at the beginning of each Type 91 record.

Offset	Name	Length	Format	Description
0 0	SMFRCD91	92	structure	Header and Self-Defining Sections
0 0	SMF91HDR	28	structure	Record header
0 0	SMF91LEN	2	binary	Record length
2 2	SMF91SEG	2	binary	Segment descriptor
4 4	SMF91FLG	1	binary	System indicator **Bit Meaning When Set** 0 Subsystem identification follows system identification 1 Subtypes used 2 Reserved 3-6 MVS Version indicators (Set by SMF) 7 Reserved.
5 5	SMF91RTY	1	binary	Record type - 91, X'5B'
6 6	SMF91TME	4	binary	Local time, in hundredths of a second, record was moved to SMF buffer.
10 A	SMF91DTE	4	packed	Date that the record was moved to the SMF buffer; form is 0cyydddF, where F is the sign
14 E	SMF91SID	4	EBCDIC	System identification (taken from SID parameter)
18 12	SMF91WID	4	EBCDIC	Subsystem identification for the BatchPipes address space (‘STC’ for Started Task)
22 16	SMF91STP	2	binary	Record Subtype **Note:** This field defines which record mapping is being used: **1** Subsystem Initialization **2** Subsystem Interval **3** Subsystem Ending **11** Pipe Connection Open **12** Pipe Interval **13** Pipe Connection Close **14** Pipe Create **15** Pipe Delete
24 18	SMF91SDL	4	binary	Length of Self-Defining Section

Self-Defining Section

This section contains the triplet fields (offset/length/number) that locate the other sections on the record. It is an extension of the record header and follows it physically on the record.

Check the “Number” field before you access a section of the record. A nonzero value means that the section exists on the record; a zero value indicates that the section does not exist on the record.

Offset	Name	Length	Format	Description
0 0	SMF91SDS	64	structure	Self-defining section
0 0	SMF91PRO	4	binary	Offset of Product Section. See page 165.
4 4	SMF91PRL	2	binary	Length of Product Section
6 6	SMF91PRN	2	binary	Number of Product Sections - “1”
8 8	SMF91SIO	4	binary	Offset of Subsystem Identification Section. See page 166.
12 C	SMF91SIL	2	binary	Length of Subsystem Identification Section
14 E	SMF91SIN	2	binary	Number of Subsystem Identification Sections - ‘1’
16 10	SMF91SCO	4	binary	Offset of Subsystem Controls Section. See page 167.
20 14	SMF91SCL	2	binary	Length of Subsystem Controls Section
22 16	SMF91SCN	2	binary	Number of Subsystem Controls Sections - ‘0’ or ‘1’
24 18	SMF91SAO	4	binary	Offset of Subsystem Activity section See page 168.
28 1C	SMF91SAL	2	binary	Length of Subsystem Activity Section
30 1E	SMF91SAN	2	binary	Number of Subsystem Activity Sections - ‘0’ or ‘1’
32 20	SMF91PIO	4	binary	Offset of Pipe Identification Section. See page 169.
36 24	SMF91PIL	2	binary	Length of Pipe Identification Section
38 26	SMF91PIN	2	binary	Number of Pipe Identification Sections - ‘0’ or ‘1’
40 28	SMF91PAO	4	binary	Offset of Pipe Activity Section. See page 170.
44 2C	SMF91PAL	2	binary	Length of Pipe Activity Section
46 2E	SMF91PAN	2	binary	Number of Pipe Activity Sections - ‘0’ or ‘1’
48 30	SMF91ICO	4	binary	Offset to writer connection section from start of record. See page 171.
52 34	SMF91ICL	2	binary	Length of Writer Connection Section
54 36	SMF91ICN	2	binary	Number of Writer Connection Sections - ‘0’ to ‘n’
56 38	SMF91OCO	4	binary	Offset to first reader connection section from start of record. See page 173.
60 3C	SMF91OCL	2	binary	Length of Reader Connection Section
62 3E	SMF91OCN	2	binary	Number of Reader Connection Sections - ‘0’ to ‘n’

Product Section

This section provides some general identification information of BatchPipes and the system at the time that the record was generated.

Triplet Information: Use the following triplet fields in the header/self-defining section to locate the record.

Offset SMF91PRO
Length SMF91PRL
Number SMF91PRN - set to “1” on each Type 91 record.

Offset	Name	Length	Format	Description
0 0	SMF91PR	44	structure	Product section
0 0	SMF91PNM	10	char	Product Name - ‘MVS/Pipes’
10 A	SMF91RVN	2	binary	Record Version Number - ‘1’
12 C	SMF91OSL	8	char	MVS system level (for example: ‘SP 4.2.0’)
20 14	SMF91SYN	8	char	MVS system name (SYSNAME from IEASYSxx)
28 1C	SMF91INT	8	binary	Interval timestamp (STCK format of GMT), 0 for non-interval records
36 24	SMF91INO	8	binary	Local time offset for SMF91INT field (STCK format), 0 for non-interval records

Note: Use the following fields only if the length (SMF91PRL) of the section is greater than 44 bytes.

44 2C	SMF91IST	4	binary	Interval start time (local, in .01 seconds since midnight)
48 30	SMF91ISD	4	packed	Interval start date (in the form 0cyydddF, where ‘F’ is the sign)
52 34	SMF91IET	4	binary	Interval end time (local, in .01 seconds since midnight). Note that this time may differ from the value in SMF91INT.
56 30	SMF91IED	4	packed	Interval end date (in the form 0cyydddF, where ‘F’ is the sign)

Subsystem Identification Section

This section provides general identification and parameter values for the specified BatchPipes subsystem. It appears only on subsystem-related records (Subtypes 1, 2, and 3).

Triplet Information: Use the following triplet fields in the header/self-defining section to locate the record.

Offset	SMF91SIO
Length	SMF91SIL
Number	SMF91SIN

Offset	Name	Length	Format	Description
0 0	SMF91SI	14	structure	Subsystem Identification Section
0 0	SMF91SNM	4	char	Subsystem name - from Start command
4 4	SMF91SCC	8	char	Command Control Character, blank if none specified
12 C	SMF91SST	1	binary	Subsystem Status **bit** **meaning when set** **0** Active **1** Starting **2** Stopping
12 D	SMF91STR	1	binary	Type of trace active **bit** **meaning when set** **0** Error (default) **1** Function **2** Flow

Subsystem Controls Section

This section provides general parameter values for the specified BatchPipes Subsystem. It appears only on subsystem-related records (Subtypes 1, 2, 3 and 4).

Triplet Information: Use the following triplet fields in the header/self-defining section to locate the record.

Offset	SMF91SCO
Length	SMF91SCL
Number	SMF91SCN

Offset	Name	Length	Format	Description
0 0	SMF91SC	56	structure	Subsystem controls section
0 0	SMF91SCT	4	binary	Time, in hundredths of a second, when status changed
4 4	SMF91SCD	4	packed	Date that the status changed: form is 0cyydddF, where F is the sign
8 8	SMF91SRI	4	binary	SMF subsystem recording interval, in seconds
12 C	SMF91SMP	4	binary	Reserved — Set to “125.”
16 10	SMF91SMC	4	binary	Reserved — Set to “250.”
20 14	SMF91SMD	4	binary	Reserved — Set to “0.”
24 18	SMF91SMS	4	binary	Reserved — Set to “0”
28 1C	SMF91SMV	4	binary	Reserved — Set to “0”
32 20	SMF91SOI	4	binary	Reserved — Set to “15”
26 24	SMF91SWI	4	binary	Reserved — Set to “15”
40 28	SMF91SII	4	binary	Reserved — Set to “15”
44 2C	SMF91CP1	2	binary	Reserved — Set to “80”
46 2E	SMF91CP2	2	binary	Reserved — Set to “90”
48 30	SMF91CP3	2	binary	Reserved — Set to “100”
50 32	SMF91CR1	2	binary	Reserved — Set to “80”
52 34	SMF91CC2	2	binary	Reserved — Set to “90”
54 36	SMF91CC3	2	binary	Reserved — Set to “100”

Subsystem Activity Section

This section provides usage statistics for the specified BatchPipes subsystem. It appears only on subsystem-related records (Subtypes 1, 2 and 3).

Triplet Information: Use the following triplet fields in the header/self-defining section to locate the record.

Offset	SMF91SAO
Length	SMF91SAL
Number	SMF91SAN

Offset	Name	Length	Format	Description
0 0	SMF916A	44	structure	Subsystem activity section
0 0	SMF91SPC	4	binary	Number of pipes created
4 4	SMF91SPD	4	binary	Number of pipes deleted
8 8	SMF91SPA	4	binary	Number of pipes active
12 C	SMF91SCP	4	binary	Number of connections opened
16 10	SMF91SCS	4	binary	Number of connections closed
20 14	SMF91SCA	4	binary	Number of connections active
24 18	SMF91SDI	8	long floating point	Number of bytes transferred - writer connections
32 20	SMF91SDO	8	long floating point	Number of bytes transferred - reader connections
40 24	SMF91SP1	4	binary	Reserved — Set to “0”
44 28	SMF91SP2	4	binary	Reserved — Set to “0”
48 30	SMF91SP3	4	binary	Reserved — Set to “0”
52	SMF91SC1	4	binary	Reserved — Set to “0”
56 38	SMF91SC2	4	binary	Reserved — Set to “0”
60 3C	SMF91SC3	4	binary	Reserved — Set to “0”

Pipe Identification Section

This section provides the identification for the pipe that this record represents. It appears only on pipe-related records (Subtypes 11, 12, 13, 14 and 15)

Triplet Information: Use the following triplet fields in the header/self-defining section to locate the record.

Offset	SMF91PIO
Length	SMF91PIL
Number	SMF91PIN

Offset	Name	Length	Format	Description
0 0	SMF91PI	60	structure	Pipe identification section
0 0	SMF91PIP	44	EBCDIC	Pipename
44 2C	SMF91PRF	1	binary	Record format **Bit** **Meaning When Set** **0** Fixed **1** Variable **2** Reserved **3** Blocked **4** Spanned **5** CC = ANSI **6** CC = Machine **7** Unused **Note:** For RECFM=U, Bits 0 and 1 are both set
45 2D	SMF91PMF	2	binary	Type of I/O macro instruction and options. Set to “0.” **Note:** Field mapped by DCBMACRF
47 2F		1		Reserved
48 30	SMF91PLR	4	binary	Logical record length
52 34	SMF91PBK	4	binary	Blocksize
56 38	SMF91PDE	4	binary	Pipe depth limit, number of records (if RECFM=F|V), or number of blocks (if RECFM=FB|VB) allowed in single pipe until full condition encountered.

Pipe Activity Section

This section provides the summary usage statistics for the pipe that transfers this record. The section appears on pipe-related records (Subtypes 11, 12, 13 and 15) only if there is data SMF to report. The section does not appear on the Pipe Creation record (Subtype 14).

Triplet Information: Use the following triplet fields in the header/self-defining section to locate the record.

Offset	SMF91PAO
Length	SMF91PAL
Number	SMF91PAN

Offset	Name	Length	Format	Description
0 0	SMF91PA	24	structure	Pipe activity section
0 0	SMF91POI	4	binary	Number of opens for writers
4 4	SMF91PCI	4	binary	Number of closes for writer connections
8 8	SMF91PIC	4	binary	Number of active writer connections to pipe
12 C	SMF91POO	4	binary	Number of opens for readers
16 10	SMF91PCO	4	binary	Number of closes of reader connections
20 14	SMF91POC	4	binary	Number of active reader connections to pipe

Pipe Writer Connection Section

This section provides the specific connection information for each writer to the pipe. The section appears on pipe-related records (Subtypes 11, 12, 13 and 15) only if there is data for SMF to report. The section does not appear on the Pipe Creation record (Subtype 14).

Triplet Information: Use the following triplet fields in the header/self-defining section to locate the record.

Offset	SMF91ICO
Length	SMF91ICL
Number	SMF91ICN

The number of sections on the record vary based on the number of writers to this pipe that have been defined. Multiple connections to the same pipe by a given address space result in multiple connection sections on the record for that address space. For an OPEN or CLOSE record there will be only one connection section on the record.

Offset	Name	Length	Format	Description
0 0	SMF91IC	108. 124	structure	Writer Connection Section
0 0	SMF91IJN	8	EBCDIC	Job name
8 8	SMF91IJJ	8	EBCDIC	JES job number
16 10	SMF91IRT	4	binary	Time reader recognized the JOB card for this job, in hundredths of a second.
20 14	SMF91IJD	4	packed	Date reader recognized the JOB card for this job; form is 0cyydddF where F is the sign.
24 18	SMF91ISN	8	EBCDIC	Step name
32 20	SMF91ISR	4	binary	Step number
36 24	SMF91IDD	8	EBCDIC	DD Name
44 2C	SMF91IAT	4	binary	Local time, in hundredths of a second, when the pipe was allocated for this connection.
48 30	SMF91IAD	4	packed	Date pipe connection was allocated; form is 0cyydddF where F is the sign.
52 34	SMF91IJT	4	binary	Local time, in hundredths of a second, when the connection OPEN was requested.
56 38	SMF91IRD	4	packed	Date pipe connection OPEN was requested; form is 0cyydddF where F is the sign.
60 3C	SMF91IOT	4	binary	Local time, in hundredths of a second, when the pipe connection OPEN completed.
64 40	SMF91IOD	4	packed	Date pipe connection OPEN completed; form is 0cyydddF where F is the sign.
68 44	SMF91ICT	4	binary	Time, in hundredths of a second, when the pipe was closed for this connection.
72 48	SMF91ICD	4	packed	Date pipe connection was closed; form is 0cyydddF where F is the sign.
76 4C	SMF91INA	4	binary	Number of reads (records if not blocked, blocks if blocked)

Offset	Name	Length	Format	Description
80 50	SMF91IBT	8	long floating point	Number of bytes read
88 58	SMF91IW	4	binary	Number of waits on empty
92 5C	SMF91IWT	4	binary	Total time, in hundredths of a second, connection has been waiting; readers wait on empty pipe.
96 60	SMF91IWB	4	binary	Number of waits on busy
100 64		4	binary	Reserved
104 68	SMF91IEC	4	binary	Error code, if any

Note: Use the following fields only if the length (SMF91ICL) of the section is greater than 108 bytes.

108 6C	SMF91IFT	4	binary	Time, in hundredths of a second, of the FIRST access to the pipe for this connection.
112 70	SMF91IFD	4	binary	Date of the FIRST access to the pipe for this connection; form is 0cyydddF where F is the sign.
116 74	SMF91ILT	4	binary	Time, in hundredths of a second, of the LAST access to the pipe for this connection.
120 78	SMF91ILD	4	binary	Date of the LAST access to the pipe for this connection; form is 0cyydddF where F is the sign.

Pipe Reader Connection Section

This section provides the specific connection information for each reader from the pipe. It appears on pipe-related records (Subtypes 11, 12, 13 and 15) only if there is any data for SMF to report. The section does not appear on the pipe Creation record (Subtype 14).

Triplet Information: Use the following triplet fields in the header/self-defining section to locate the record.

Offset	SMF91OCO
Length	SMF91OCL
Number	SMF91OCN

The number of sections on the record vary based upon the number readers from this pipe that have been defined. Multiple connections to the same pipe by a given address space result in multiple connection sections on the record for that address space. For an OPEN or CLOSE record there will be only one connection section on the record.

Offset	Name	Length	Format	Description
0 0	SMF91OC	108. 124	structure	Reader connection section
0 0	SMF91OJN	8	EBCDIC	Job name
8 8	SMF91OJJ	8	EBCDIC	JES job number
16 10	SMF91OJT	4	binary	Time reader recognized the JOB card for this job, in hundredths of a second.
20 14	SMF91OJD	4	packed	Date reader recognized the JOB card for this job; form is 0cyydddF where F is the sign.
24 18	SMF91OSN	8	EBCDIC	Step name
32 20	SMF91OSR	4	binary	Step number
36 24	SMF91ODD	8	EBCDIC	DD Name
44 2C	SMF91OAT	4	binary	Time, in hundredths of a second, when the pipe was allocated for this connection.
48 30	SMF91OAD	4	packed	Date pipe connection was allocated; form is 0cyydddF where F is the sign.
52 34	SMF91ORT	4	binary	Time, in hundredths of a second, when the connection OPEN was requested.
56 38	SMF91ORD	4	packed	Date pipe connection OPEN was requested; form is 0cyydddF where F is the sign.
60 3C	SMF91OOT	4	binary	Local time, in hundredths of a second, when the pipe connection OPEN was completed.
64 40	SMF91OOD	4	packed	Date pipe connection OPEN completed; form is 0cyydddF where F is the sign.
68 44	SMF91OCT	4	binary	Local time, in hundredths of a second, when the pipe was closed for this connection.
72 48	SMF91OCD	4	packed	Date pipe connection was closed; form is 0cyydddF where F is the sign.
76 4C	SMF91ONA	4	binary	Number of writes (records if not blocked, blocks if blocked)

Offset	Name	Length	Format	Description
80 50	SMF91OBT	8	long floating point	Number of bytes written
88 58	SMF91OW	4	binary	Number of waits on full
92 5C	SMF91OWT	4	binary	Total time, in hundredths of a second, connection has been waiting on full pipe.
96 60	SMF91OWB	4	binary	Number of waits on busy
100 64		4	binary	Reserved
104 68	SMF91OEC	4	binary	Error code, if any

Note: Use the following fields only if the length (SMF91OCL) of the section is greater than 108 bytes.

108 6C	SMF91OFT	4	binary	Local time, in hundredths of a second, of the FIRST access to the pipe for this connection.
112 70	SMF91OFD	4	binary	Date of the first access to the pipe for this connection; form is 0cyydddF where F is the sign.
116 74	SMF91OLT	4	binary	Local time, in hundredths of a second, of the LAST access to the pipe for this connection.
120 78	SMF91OLD	4	binary	Date of the LAST access to the pipe for this connection; form is 0cyydddF where F is the sign.

Appendix B: BatchPipes Query Service

This appendix describe a General-use Programming Interface and associated guidance information.

BatchPipes provides a query service that allows you to obtain information about a particular BatchPipes subsystem. Use the query service if you want to provide a sophisticated monitor of BatchPipes performance and activity and you are familiar with using assembler language to code application programs. The name of the service is ASFPMSQI; use it to:

- Provide a real-time presentation of BatchPipes activity different from those offered by BatchPipes or other vendors.
- Flag performance problems based on historical data, or an analysis of real time trends.
- Provide timer or event-driven monitoring of BatchPipes.

Information you can obtain through the query service is:

- Subsystem state
- List of jobs using pipes
- List of opened pipes
- Status information based on job name or names
- Status information based on pipe name or names
- Status information about inactivity thresholds and exceptions

The query service consists of a callable module named ASFPMSQI, an input parameter list called the external monitor request block (XMRB), and an output parameter area called the subsystem query interface block (SQIB).

Preparing to Use the BatchPipes Query Service

To use the query service, you load and call ASFPMSQI. Module ASFPMSQI is installed in BatchPipes system library SYS1.SASFPLIB during BatchPipes initialization. The JCL that invokes the user program must include a STEPLIB or JOBLIB statement for the SYS1.SASFPLIB BatchPipes system library.

The query service requires two parameter areas, both of which you will obtain storage for: the XMRB and the SQIB.

- The format for the XMRB is in the ASFPZXMR mapping macro; the format for the SQIB is in the ASFPZSQI mapping macro. Both are Assembler H mapping macros supplied with BatchPipes.
- ASFPZXMR and ASFPZSQI are installed into system library SYS1.MODGEN during BatchPipes installation.

Your operations staff might move the module and the two macros to different libraries; contact the operations staff to locate the macros and module if they are not in the standard libraries.

Environment of the Caller

Requirements for the caller of ASFPMSQI are:

Minimum Authorization	Problem state and any PSW key
Dispatchable Unit Mode	Task
Cross Memory Mode	HASN=PASN
Amode	31-bit
ASC Mode	Primary
Interrupt Status	Enabled for I/O and external interrupts
Locks	No locks held
Control Parameters	Control parameter areas must be obtained by the caller and in the key of the caller.

Invoking the Query Services

Requesting information from the BatchPipes query service requires two separate calls to ASFPMSQI:

- First call: Initialize the XMRB and SQIB parameter areas:
  - Obtain and initialize the XMRB
  - Call ASFPMSQI to obtain the length of the SQIB
  - Obtain and initialize the SQIB

  Details about the first call begin on page 177.
- Second call: Issue the specific query service request:
  - Initialize the XMRB and SQIB before issuing the specific request
  - Call ASFPMSQI to obtain BatchPipes information

  Details about the second call begin page 178.

The example page 194 through 196 shows how to initialize the XMRB and SQIB, how to call ASFPMSQI, and also how to request job status for all jobs. You may want to refer to the example as you use the following sections.

Query Service Requests

You issue the query service request by initializing the XMRB with the symbolic name of the request. Table 176 describes each request, tells where you can find details, and shows the symbolic name you use when you issue the request.

Figure 54. Query Service Requests

Description of Request	Page	Symbolic Name
Verify state of the subsystem.	179	XMRB_SUBSYS_VERIFY
Summary information about subsystem activity.	179	XMRB_SUBSYS_STATUS
List of all active jobs.	180	XMRB_LIST_JOBS
List of all active pipe datasets.	180	XMRB_LIST_PIPES
Detailed status about selected jobs.	180	XMRB_JOBS_STATUS
Detailed status about selected pipe dataset names.	182	XMRB_PIPES_STATUS
Summary of inactivity thresholds and optionally details about threshold exceptions for selected jobs.	183	XMRB_THRESH_STATUS

First Call: Initializing the XMRB and SQIB Parameter Areas

The purpose of the first call is to obtain storage for and initialize the XMRB and SQIB. For each query service request, the steps are the same. Follow the procedures **in the order specified**.

1. Obtain storage for the XMRB in the caller's key. Some requests require job or pipe names as input. As a result, the size of the XMRB depends on the particular request, as the following table outlines:

Request	Size of XMRB in decimal bytes
XMRB_SUBSYS_VERIFY	40
XMRB_SUBSYS_STATUS	40
XMRB_LIST_JOBS	2040
XMRB_LIST_PIPES	11040
XMRB_JOBS_STATUS	2040
XMRB_PIPES_STATUS	11040
XMRB_THRESH_STATUS	2040

   The sizes for the XMRB are based on a maximum of 250 possible job or pipe names specified as input in a single XMRB.

   Alternately, if you know in advance the maximum number of jobnames or pipenames that you will specify as input in the XMRB, you can use the following formula to calculate the XMRB size in bytes:

```
XMRB size (in byte) = (maximum number of names) * (length of name) +
                      (size of XMRBHEAD)
```

   where:

   - Maximum number of names is the maximum names you expect to input
   - Length of a name is either 8 for jobname input or 44 for pipe dataset name input
   - Size of the XMRBHEAD is the header area of the XMRB (40 bytes).

2. Establish addressability to the XMRB by issuing the following:

```
USING XMRB,REG
```

   where REG is a register other than zero that contains the address of the storage obtained in step 1.

3. Clear the XMRB to binary zeros.

4. The purpose of the initial call to ASFPMSQI is to obtain the size of the SQIB so you know how much storage to obtain. Before calling ASFPMSQI, initialize the following fields in the XMRB:

Field Name	Value	Is it Required?
XMRBEYE	XMRBKEYE	Yes
XMRBVER	XMRBKVER	Yes
XMRB_SUBPOOL	Subpool number for XMRB	No
XMRB_SIZE	Size of XMRB in bytes	No
XMRB_REQUEST	XMRB_BUFF_SIZE	Yes

5. After you initialize the XMRB, issue a LOAD and then a CALL to get the length of the SQIB:

   ```
   LOAD  EP=ASFPMSQI,LOADPT=ASFPMSQI
   L     15,ASFPMSQI
   CALL  (15),(XMRBPTR)
   ```

   where XMRBPTR is the address of the XMRB.
6. When control returns, the XRMB_SQIB_BUFF_SIZE field contains the length of the SQIB in bytes.
7. Using the length returned in step 6, obtain storage for the SQIB in the key of the caller.
8. Establish addressability to the SQIB by issuing the following:

   ```
   USING SQIB,REG
   ```

   where REG is a register other than zero that contains the address of the storage obtained in step 7.
9. Clear the SQIB to binary zeros.
10. Now, initialize the following fields in the XMRB and SQIB in preparation for the second call to ASFPMSQI:

Field Name	Value	Is it Required?
SQIBEYE	SQIBKEYE	Yes
SQIB_VER	SQIBKVER	Yes
SQIB_SUBPOOL	Subpool number for SQIB	No
SQIB_SIZE	Size of SQIB obtained in bytes	Yes
SQIB_XMRB_PTR	Address of XMRB	No
SQIB_ERR_CODE	0	No
XMRB_SQIB_PTR	Address of the SQIB obtained	Yes
XMRB_REQUEST	Symbolic name of the specific request. See Figure 54 on page 176.	Yes

Second Call: Issuing the Query Request

After completing the first call, you are ready to issue the request. You issue the request by initializing the XMRB with the symbolic name associated with the request and other fields that are associated with the specified request. Figure 54 on page 176 describes each request, tells where you can find details, and shows the symbolic name you use. Some fields in the XMRB depend on which request you are issuing. For example, to ask about job status, the valid jobnames must be listed in the XMRB. The following sections describe how to issue each query service request; **they assume that you have already performed the steps described for the first call**. As part of the procedures for the first call, you loaded and saved the address for ASFPMSQI. Do not delete the module until after the final call to ASFPMSQI is complete.

For each request, be sure you code XMRB_SSNAME_REQ to reflect the name of the BatchPipes subsystem that you want information about.

Before you use the output from any request, check the SQIB fields that are listed in Figure 55.

Figure 55. Standard Output From All Requests

Field Name	Length (bytes)	Hex offset	Attribute	Description
SQIB_ERR_CODE	4	20	binary	Error code, if return code is nonzero.
SQIB_SUB_FLAGS	1	24	bit	Subsystem status flags
SQIB_F_STARTING	1	24		X'80' means subsystem is starting
SQIB_F_STARTED	1	24		X'40' means subsystem is started
SQIB_F_STOPPING	1	24		X'20' means subsystem is stopping

The sections that follow describe procedures for specifying the requests and describe information that the query service returns.

Verify Subsystem State Request

The verify subsystem state request determines whether the specified subsystem is active, stopping, not active, or does not exist. The output is one of four return codes, each of which corresponds to one of these states. Use this request before you submit requests to the query service to verify that the subsystem is valid and operational.

To issue the verify subsystem request:

1. Fill in the XMRB_REQUEST field with XMRB_SUBSYS_VERIFY
2. Invoke ASFPMSQI by issuing a CALL, as described page 178.

On return from the verify subsystem request, the return code in Register 15 describes the state of the subsystem, as in Figure 56:

Figure 56. Return Codes from the Verify Subsystem State Request

Return Code	The state of the subsystem
0	The subsystem is an active BatchPipes subsystem.
4	The subsystem is stopping.
8	The subsystem is not active.
12	The subsystem does not exist

Subsystem Status Request

The subsystem status request returns summary information about the specified subsystem. Information such as the number of jobs, number of pipes, number of readers and writers, maximum idle and wait times, and summary threshold information is returned in the SQIB. To request subsystem status:

1. Fill in the XMRB_REQUEST field with XMRB_SUBSYS_STATUS.
2. Invoke ASFPMSQI by issuing a CALL.

The output information from the subsystem status request is in the SQIB in the SQIB_SUB_STATUS section. A full mapping of the SQIB is in “The SQIB Output Parameter Area” on page 188.

On return from the subsystem status request, Register 15 contains a return code. The valid return codes and suggested actions are as follows:

Figure 57. Return Codes from Subsystem Status Request

Return Code	Description	Suggested action
0	Successful completion	None
4	Error processing request	Check SQIB_ERR_CODE field for error code and "Error Codes" on page 184.
8	The XMRB parameter area is invalid	Ensure that the eye catcher (XMRBEYE) and version number (XMRBVER) fields are correct.

Lists of All Jobs and Pipes Requests

The list jobs and list pipes requests are similar. The list jobs request returns a list of all active jobs; the list pipes request returns a list of all pipe data sets. In both cases, duplicate names are removed from the list. To issue the request:

1. Fill in XMRB_REQUEST field with XMRB_LIST_JOBS or fill in XMRB_REQUEST field with XMRB_LIST_PIPES
2. Invoke ASFPMSQI by issuing a CALL.

The output of the list jobs or list pipes request is an array of job names or pipe names. The array, count of elements, and length per entry are returned in the SQIB, as the following table summarizes:

Figure 58. Output of List Jobs and List Pipes Requests

Field Name	Length (bytes)	Hex offset	Attribute	Description
SQIB_ENTRY_COUNT	4	14	binary	Number of array entries returned in output
SQIB_ENTRY_LENGTH	4	18	binary	Length of each entry in output array
SQIB_JOBNAME or SQIB_PIPENAME	variable	100	array	List of all active (8-character) jobnames or (44-character) pipenames.

On return, Register 15 contains a return code. Valid return codes and suggested actions are the same as those for the subsystem status request. See Figure 57 on page 180.

Jobs Status Request

The jobs status request returns the status of a specified group of jobs. Information such as associated pipe name, wait/idle/wait-for-open time, and blocks transferred is returned in the SQIB. In XMRB fields, you specify the names of the jobs for which you request status and other information, as described in the following table:

Figure 59. Input to the Jobs Status Request

Initialize these fields:	Length (bytes)	HEX offset	Attribute	Description
XMRB_SSNAME_REQ	4	8	char	Name of BatchPipes subsystem
XMRB_REQUEST	4	10	binary	Fill in with XMRB_JOBS_STATUS
XMRB_ENTRY_COUNT	4	14	binary	Number of jobnames for which you request status
XMRB_ENTRY_LENGTH	4	18	binary	Set the field to the value 8, the maximum length of a jobname
XMRB_JOBNAME	Variable	28	array	List of jobnames for which you request status

You must initialize XMRB_JOBNAME with up to 250 names of selected jobs. The number of names you specify in the list must match the value in XMRB_ENTRY_COUNT.

Those names might include wild card characters, described as follows:

Specifying Jobnames and Pipenames

Because input to the XMRB is a list of names of specific jobs and pipes, the size of the XMRB varies. Each jobname is 8 characters in length and each pipename is 44 characters in length; in both cases, the names can consist of wildcard characters. Specify each name in one of the following ways:

- A string of characters padded to the right with blanks
- A string of characters containing any combination of two special characters:
  - Question Mark (?) to represent a single-character wild card
  - Asterisk (*) to represent a multiple-character wild card
- A single asterisk padded to the right with blanks to indicate that you request status for all jobs or pipes controlled by the specified subsystem

Invoke ASFPMSQI by issuing a CALL.

On return, the SQIB contains the requested information, as the following table describes. Note that part of the output in the SQIB is an array of job status entries.

Figure 60. Output of Job Status Request

Field Name	Length (bytes)	Hex offset	Attribute	Description
SQIB_ENTRY_COUNT	4	14	binary	Number of unique job status array entries returned in output.
SQIB_ENTRY_LENGTH	4	18	binary	Length of each entry in bytes
SQIB_STATUS_LIST	variable	100	array	Array of job status entries.

Because one job can connect to many pipes, SQIB_ENTRY_COUNT reflects the number of unique job/pipe combinations that match the names you specified in the XMRB_JOBNAME array.

Figure 67 on page 192 describes the format of each entry in the array and gives the offset of the field within the SQIB_STATUS_LIST mapping.

On return from the jobs status request, Register 15 contains a return code. Valid return codes and suggested actions are the same as those for the subsystem status request. See Figure 57 on page 180.

Pipes Status Request

The pipes status request returns the status of a specified group of pipes. Information such as associated job name, wait/idle/wait-for-open time, and blocks transferred is returned in the SQIB. Pipes for which status is requested must be specified in the XMRB. The pipes status request returns the same information as the jobs status request; use the pipes status request if you know pipenames, rather than jobnames.

The following table describes how to update the XMRB before you issue the pipes status request:

Figure 61. Input to the Pipes Status Request

Initialize these fields:	Length (bytes)	HEX offset	Attribute	Description
XMRB_SSNAME_REQ	4	8	char	Name of BatchPipes subsystem
XMRB_REQUEST	4	10	binary	Fill in with XMRB_PIPES_STATUS
XMRB_ENTRY_COUNT	4	14	binary	Number of pipenames for which you request status
XMRB_ENTRY_LENGTH	4	18	binary	Set the field to the value 44, the maximum length of a pipe dataset name
XMRB_PIPENAMES	Variable	28	array	Array of pipenames for which you request status

Initialize XMRB_PIPENAMES with the 44-character names of the selected pipes. The size of the XMRB varies; it contains names that might include wild card characters. For help coding the names, see “Specifying Jobnames and Pipenames” page 181.

The number of names you specify in the array must match the value in XMRB_ENTRY_COUNT.

Invoke ASFPMSQI by issuing a CALL. On return, the SQIB contains the requested information, as the following table describes. Note that part of the output in the SQIB is an array of status entries.

Figure 62. Output of Pipe Status Request

Field Name	Length (bytes)	Hex offset	Attribute	Description
SQIB_ENTRY_COUNT	4	14	binary	Number of unique pipe status array entries returned in output
SQIB_ENTRY_LENGTH	4	18	binary	Length of each entry in bytes
SQIB_STATUS_LIST	variable	100	array	Array of pipe status information

Notes on understanding the output of the pipe status request are:

- Because one job can connect to many pipes, SQIB_ENTRY_COUNT reflects the number of unique job/pipe combinations that match the names you specified in the XMRB_PIPENAMES array.
- The number of entries in the SQIB_STATUS_LIST output array is equal to SQIB_ENTRY_COUNT, and each entry is X'6C' bytes long. The format of the entry is in Figure 67 on page 192.

On return from the pipes status request, Register 15 contains a return code. Valid return codes and suggested actions are the same as those for the subsystem status request. See Figure 57 on page 180.

Threshold Status Request

The threshold status request returns a threshold exception summary for the subsystem and, optionally, detailed information on the specific jobs that are threshold exceptions. If you request job-related threshold data, two pieces of information need to be present in the XMRB.

- The specific jobs for which you request threshold information
- The type or types of exceptions that you want.

Requesting job-specific threshold information is optional; and the job-specific information is returned **in addition to** the threshold summary data. The following procedures outline the steps required to obtain threshold status information:

1. Fill in XMRB_REQUEST with XMRB_THRESH_STATUS
2. Invoke ASFPMSQI by issuing a CALL.

If you also want to request job-specific threshold information, the following table lists the fields you must initialize prior to calling ASFPMSQI:

Figure 63 (Page 1 of 2). Input to the Threshold Status Request

Initialize these fields:	Length (bytes)	HEX offset	Attribute	Description
XMRB_SSNAME_REQ	4	8	char	Name of BatchPipes subsystem
XMRB_REQUEST	4	10	binary	Fill in with XMRB_THRESH_STATUS
XMRB_ENTRY_COUNT	4	14	binary	Number of jobnames for which you request threshold status
XMRB_ENTRY_LENGTH	4	18	binary	Set the field to the value 8, the maximum length of a jobname
XMRB_THRESH_FILTER	4	24	8-bit	Selection for threshold type
XMRB_RDR_JOBS	1	24	bit	X'80' means select all reader threshold exception jobs

Figure 63 (Page 2 of 2). Input to the Threshold Status Request

Initialize these fields:	Length (bytes)	HEX offset	Attribute	Description
XMRB_WTR_JOBS	1	24	bit	X'40' means select all writer threshold exception jobs
XMRB_IDLE_JOBS	1	24	bit	X'20' means select all idle threshold exception jobs
XMRB_WFO_JOBS	1	24	bit	X'10' means select all wait-for-open threshold exception jobs
XMRB_WAIT_JOBS	1	24	bit	X'08' means select all waiting threshold exception jobs
XMRB_JOBNAME	Variable	28	array	Array of jobnames for which you request status

Notes on coding the threshold status request are:

- Initialize the XMRB_JOBNAME array with the names of the selected jobs. The XMRB contains an array of varying size that contains the names. Jobnames may include wild card characters. See "Specifying Jobnames and Pipenames" on page 181.
- Specify any combination of reader/writer names and threshold types in the XMRB. If you want all types of exceptions matching the jobnames, turn on each of the following bits:
  - XMRB_RDR_JOBS
  - XMRB_WTR_JOBS
  - XMRB_IDLE_JOBS
  - XMRB_WFO_JOBS
  - XMRB_WAIT_JOBS
- The number of names you specify in the array must match the value in XMRB_ENTRY_COUNT.

Invoke ASFPMSQI by issuing a CALL.

On return, the SQIB contains the requested threshold summary information in the SQIB_THRESHOLD_STATUS section. A full mapping of the SQIB is in "The SQIB Output Parameter Area" on page 188. Note that this information is output whether or not you requested job-specific threshold information. Job-specific information is found in the SQIB_THRESH_LIST array, where each entry contains details about a specific job that is an exception.

The SQIB_THRESH_LIST is valid only if job-specific threshold information is requested and the jobnames input in the XMRB matches the existing threshold exceptions. Each entry in the array is X'40' byte long; SQIB_ENTRY_COUNT gives the total number of entries in the array.

On return from the threshold status request, Register 15 contains a return code. Valid return codes and suggested actions are the same as those for the subsystem status request. See Figure 57 on page 180.

Error Codes

Figure 64 documents the error codes that can occur when ASFPMSQI returns with a return code of 04 in GPR15. "**Symbol**" gives the symbolic name for the different error code values. These symbols can be found in macro ASFPZXMR.

Figure 64 (Page 1 of 2). Error Codes for Return Code 04

Error Code Decimal	Meaning and Action
21	**Symbol**: XMRB_RC_INVALID_XMRB **Meaning**: XMRB eye catcher or version incorrect. • Fields XMRBEYE and/or XMRBVER not correctly initialized. • Pointer to XMRB is invalid.
22	**Symbol**: XMRB_RC_INVALID_SQIB **Meaning**: SQIB eye catcher or version incorrect. • Fields SQIBEYE and/or SQIB_VER not correctly initialized. • Pointer to XMRB is invalid.
23	**Symbol**: XMRB_RC_INVALID_ENTRY_LENGTH **Meaning**: The value of the entry length specified is not 8 characters for job names or 44 (X'2C') characters for pipe data set names.
24	**Symbol**: XMRB_RC_INVALID_COUNT **Meaning**: The XMRB_ENTRY_COUNT value is less than or equal to zero or greater than the maximum connections allowed by BatchPipes.
31	**Symbol**: XMRB_RC_INVALID_BPCB1 **Meaning**: Subsystem name is valid, but: • Is not a valid BatchPipes subsystem. • Is being initialized. • An error has occurred with the BatchPipes subsystem. Validate that the subsystem name is a BatchPipes subsystem, and if possible that the subsystem is operational. If this error occurs repeatedly, contact the system programmer.
32	**Symbol**: XMRB_RC_NO_BPCB1 **Meaning**: Subsystem name is valid, but: • Is not a valid BatchPipes subsystem. • Is being initialized. • An error has occurred with the BatchPipes subsystem. Validate that the subsystem name is a BatchPipes subsystem, and if possible that the subsystem is operational. If this error occurs repeatedly, contact the system programmer.
33	**Symbol**: XMRB_RC_NO_BPCB2 **Meaning**: Subsystem name is valid, but: • Is not a valid BatchPipes subsystem. • Is being initialized. • An error has occurred with the BatchPipes subsystem. Validate that the subsystem name is a BatchPipes subsystem, and if possible that the subsystem is operational. If this error occurs repeatedly, contact the system programmer.
34	**Symbol**: XMRB_RC_BPMVS_DISABLED **Meaning**: A valid BatchPipes subsystem name was requested, but the subsystem is no longer active. Possible reasons are: • BatchPipes was stopped, forced, or abended. • BatchPipes was not restarted after being brought down.
35	**Symbol**: XMRB_RC_BPMVS_STOPPING **Meaning**: A valid BatchPipes subsystem name was requested, but the subsystem is stopping.
36	**Symbol**: XMRB_RC_THRESH_INACTIVE **Meaning**: BatchPipes threshold monitoring is not active. Thresholds were turned off due to a BatchPipes failure.

Figure 64 (Page 2 of 2). Error Codes for Return Code 04

Error Code Decimal	Meaning and Action
40	**Symbol**: XMRB_RC_BUFF_FULL **Meaning**: No more space is available in the SQIB for the next response. • Too small a data area was obtained for the SQIB. • The SQIB_SIZE was initialized incorrectly or changed. • The SQIB size was not obtained through the XMRB_BUFF_SIZE request according to procedures described page 177.
44	**Symbol**: XMRB_RC_BUFF_LENGTH **Meaning**: The SQIB data area is too small for any response. • Too small a data area was obtained for the SQIB. • The SQIB_SIZE was initialized incorrectly or changed. • The SQIB size was not obtained by using the XMRB_BUFF_SIZE request according to procedures described page 177.
45	**Symbol**: XMRB_RC_INCOMPLETE_OUTPUT **Meaning**: ASFPMSQI was unable to obtain all the data requested. Data collected up to the 'failure' is returned to the caller. The request is automatically retried a number of times before it returns this error code. Try resubmitting the request. If the failure continues to occur, contact the IBM support center. Possible reasons for this error are: • The BatchPipes subsystem has a lot of activity, such as pipe connection allocations and unallocations. • Jobs using BatchPipes are ending, abending, or being cancelled with high frequency
46	**Symbol**: XMRB_RC_PIPES_UNAVAILABLE **Meaning**: ASFPMSQI was unable to collect the data requested. You can resubmit the request to ASFPMSQI. If the failure occurs repeatedly, contact the IBM support center. Possible reasons for the error are: • The BatchPipes subsystem has a lot of activity, such as pipe connection allocations and unallocations. • Jobs using BatchPipes are ending, abending, or being cancelled with high frequency
99	**Symbol**: XMRB_RC_FUNCTION_NOT_SUPPORTED **Meaning**: Input in XMRB_REQUEST field is not valid.

Query Service Usage Notes

This section presents additional information on using the BatchPipes query service.

1. The query service request might return a SQIB_ERR_CODE of 45 (incomplete output) or 46 (BatchPipes subsystem is not available). A likely reason for these errors is that the query service tries to gather information without interfering or stopping ongoing BatchPipes subsystem activity. As a result, if there is a high level of BatchPipes job and pipe activity, the query service may fail. In this case, you can resubmit the request. If requests continue to fail, contact the IBM support center.
2. Once the data is returned to the program, you can reuse the XMRB and SQIB parameter areas, keeping the following in mind:
   - Clear the XMRB and SQIB to binary zeros after each request.
   - Each XMRB should point to its corresponding SQIB.
   - The process for initializing the XMRB and SQIB for each request is no different when you reuse the parameter areas.
3. Before you use any of the information provided by the query service, check the following data:

- The SQIB_SUB_FLAGS field to make sure the subsystem being queried is not in the process of starting or stopping.
- The return code, to make sure it is valid.
- SQIB_ERR_CODE field if a return code of 4 is received; see “Error Codes” on page 184.

4. The query service does not check a caller's authorization to request certain information. Therefore, any authorization checking must occur within your programs.
5. The BatchPipes subsystem name must be specified as input for each service request. You may either use the information in “ Appendix C: BatchPipes Version Information” on page 197 or contact your operations staff to determine the names of the valid BatchPipes subsystems.
6. The data produced by the BatchPipes query service is a snapshot of ongoing BatchPipes activity. Some data that is collected may have changed slightly between the time you initiate a request and the time it is output in the SQIB. This behavior is normal; the information is substantially correct.

XMRB Input Parameter Area

Figure 65 (Page 1 of 2). XMRB Input Parameter Area

Hex Offset	Field Name	Length	Attribute	Description
0	XMRBHEAD	0		Header section of the mapping
0	XMRBEYE	4	char	Eye-catcher to be set to XRMBKEYE
4	XMRBVER	1	8 bit	Version number to be set to XMRBKVER
8	XMRB_SSNAME_REQ	4	char	BatchPipes subsystem for request
C	XMRB_SUBPOOL	1	8 bit	XMRB subpool number
D	XMRB_SIZE	3	binary	Size of XMRB buffer in bytes
10	XMRB_REQUEST	4	binary	Symbolic name of the request; see Figure 54 on page 176 for a list of valid requests.
14	XMRB_ENTRY_COUNT	4	binary	Number of entries in the input jobname or pipename array
18	XMRB_ENTRY_LENGTH	4	binary	Length of an entry (either jobname length or pipename length)
1C	XMRB_SQIB_PTR	4	binary	Address of the SQIB output response buffer
20	XMRB_SQIB_BUFF_SIZE	4	binary	Maximum size of the SQIB output response buffer
24	XMRB_THRESH_FILTER	8	bit	Threshold exception selection, where each bit represents a type of threshold exception that is to be in the requested list.
24	XMRB_RDR_JOBS	1		X'80' means include threshold exception for all reader jobs
24	XMRB_WTR_JOBS	1		X'40' means include threshold exception for all writer jobs

Figure 65 (Page 2 of 2). XMRB Input Parameter Area

Hex Offset	Field Name	Length	Attribute	Description
24	XMRB_IDLE_JOBS	1		X'20' means include threshold exception all idle jobs.
24	XMRB_WFO_JOBS	1		X'10' means include threshold exception for all waiting-for-open jobs
24	XMRB_WAIT_JOBS	1		X'08' means include threshold exception for all waiting jobs
28	XMRB_DATA_ARRAY	variable	char	An array of jobnames or pipenames.
28	XMRB_PIPENAMES	array	char	An array of pipenames; each pipename is 44 (decimal) bytes long.
28	XMRB_JOBNAME	array	char	An array of jobnames; each jobname is 8 characters long
28	XMRB_SSN	array	char	Reserved

The SQIB Output Parameter Area

Figure 66 (Page 1 of 4). SQIB Output Parameter Area

Hex Offset	Field Name	Length	Attribute	Description
0	SQIB_HEADER	0		Header section of the mapping
0	SQIBEYE	4	char	Eye catcher, see SQIBKEYE
4	SQIB_VER	1	8 bit	Version number to be set to SQIBKVER
8	SQIB_SSNAME_REQ	4	char	Subsystem name for the request
C	SQIB_SUBPOOL	1	8 bit	Subpool number of SQIB
D	SQIB_SIZE	3	binary	Size of SQIB in bytes
10	SQIB_REQUEST	4	binary	Symbolic name of the request
14	SQIB_ENTRY_COUNT	4	binary	Number of entries being returned if the output is a variable size array.
18	SQIB_ENTRY_LENGTH	4	binary	Size of each entry in the array if the output is a variable size array.
1C	SQIB_XMRB_PTR	4	binary	Pointer to the XMRB associated with this request.
20	SQIB_ERR_CODE	4	binary	Error code associated with a return code of 4.
24	SQIB_SUB_FLAGS	1	bit	Subsystem status flags
24	SQIB_F_STARTING	1		X'80' means subsystem is starting
24	SQIB_F_STARTED	1		X'40' means subsystem is started

Figure 66 (Page 2 of 4). SQIB Output Parameter Area

Hex Offset	Field Name	Length	Attribute	Description
24	SQIB_F_STOPPING	1		X'20' means subsystem is stopping
28	SQIB_SUB_STATUS	X'70'	char	Subsystem status mapping
28	SQIB_SUB_JOBS	4	binary	Number of unique jobs using BatchPipes
2C	SQIB_SUB_PIPES	4	binary	Number of open pipe datasets
30	SQIB_SUB_ALLOCS	4	Binary	Number of connections having completed allocation
34	SQIB_SUB_OPENS	4	binary	Number of connections having completed open
38	SQIB_SUB_CONNECT	4	binary	Total number of connections
3C	SQIB_SUB_WTRS	4	binary	Number of writer connections
40	SQIB_SUB_WTRS_IDLE	4	binary	Number of writers idle
44	SQIB_SUB_WTRS_WFO	4	binary	Number of writers waiting for open
48	SQIB_SUB_WTRS_WAIT	4	binary	Number of writers waiting on full pipe
4C	SQIB_SUB_WTRS_MAXWAIT	4	binary	Maximum time a writer has waited on a full pipe
50	SQIB_SUB_WTRS_MAXIDLE	4	binary	Maximum time a writer has been idle
54	SQIB_SUB_WTRS_MAXWFO	4	binary	Maximum time a writer has been in wait-for-open
58	SQIB_SUB_WTRS_STARTED	4	binary	Writer connections where at least one block transfer completed
5C	SQIB_SUB_RDRS	4	binary	Number of reader connections
60	SQIB_SUB_RDRS_IDLE	4	binary	Number of readers idle
64	SQIB_SUB_RDRS_WFO	4	binary	Number of readers waiting for open
68	SQIB_SUB_RDRS_WAIT	4	binary	Number of readers waiting on empty pipe
6C	SQIB_SUB_RDRS_MAXWAIT	4	binary	Maximum time a reader has waited on an empty pipe
70	SQIB_SUB_RDRS_MAXIDLE	4	binary	Maximum time a reader has been idle
74	SQIB_SUB_RDRS_MAXWFO	4	binary	Maximum time a reader has been in wait-for-open
78	SQIB_SUB_RDRS_STARTED	4	binary	Reader connections where at least one block transfer completed
7C	SQIB_SUB_TIME_OVERFLOW	1	bit	Indicates whether any of the MAXTIME values have exceeded 1 year.

Figure 66 (Page 3 of 4). SQIB Output Parameter Area

Hex Offset	Field Name	Length	Attribute	Description
7C	SQIB_SUB_R_MWAIT_EXCD	1		X'80' means SQIB_SUB_RDRS_MAXWAIT field has exceeded 1 year and is invalid.
7C	SQIB_SUB_R_MIDLE_EXCD	1		X'40' means SQIB_SUB_RDRS_MAXIDLE field has exceeded 1 year and is invalid.
7C	SQIB_SUB_R_MWFO_EXCD	1		X'20' means SQIB_SUB_RDRS_MAXWFO field has exceeded 1 year and is invalid.
7C	SQIB_SUB_W_MWAIT_EXCD	1		X'10' means SQIB_SUB_WTRS_MAXWAIT field has exceeded 1 year and is invalid.
7C	SQIB_SUB_W_MIDLE_EXCD	1		X'08' means SQIB_SUB_WTRS_MAXIDLE field has exceeded 1 year and is invalid.
7C	SQIB_SUB_W_MWFO_EXCD	1		X'04' means SQIB_SUB_WTRS_MAXWFO field has exceeded 1 year and is invalid.
80	SQIB_SUB_THR_RIDLE_CEXCP	4	binary	Number of idle reader threshold exceptions
84	SQIB_SUB_THR_RWFO_CEXCP	4	binary	Number of wait-for-open reader threshold exceptions
88	SQIB_SUB_THR_RWAIT_CEXCP	4	binary	Number of waiting on empty reader threshold exceptions
8C	SQIB_SUB_THR_WIDLE_CEXCP	4	binary	Number of idle writer exceptions
90	SQIB_SUB_THR_WWFO_CEXCP	4	binary	number of wait-for-open writer threshold exceptions
94	SQIB_SUB_THR_WWAIT_CEXCP	4	binary	Number of waiting on full writer threshold exceptions
98	SQIB_THRESHOLD_STATUS	X'68'	char	Threshold status mapping
98	SQIB_THRESH_MAXPIPE	4	binary	The maximum number of pipes allowed per subsystem
9C	SQIB_THRESH_MAXCONN	4	binary	The maximum number of job connections allowed per subsystem
A0	SQIB_THRESH_PIPE1	X'A0'	binary	Reserved
A4	SQIB_THRESH_PIPE2	X'A4'	binary	Reserved
A8	SQIB_THRESH_PIPE3	X'A8'	binary	Reserved
AC	SQIB_THRESH_CONN1	X'AC'	binary	Reserved
B0	SQIB_THRESH_CONN2	X'B0'	binary	Reserved

Figure 66 (Page 4 of 4). SQIB Output Parameter Area

Hex Offset	Field Name	Length	Attribute	Description
B4	SQIB_THRESH_CONN3	X'B4'	binary	Reserved
B8	SQIB_THRESH_RIDLE_TIME	4	binary	Reader idle threshold setting in minutes.
BC	SQIB_THRESH_RWAIT_TIME	4	binary	Reader wait threshold setting in minutes.
C0	SQIB_THRESH_RWFO_TIME	4	binary	Reader wait-for-open threshold setting in minutes.
C4	SQIB_THRESH_WIDLE_TIME	4	binary	Writer idle threshold setting in minutes.
C8	SQIB_THRESH_WWAIT_TIME	4	binary	Writer wait threshold setting in minutes.
CC	SQIB_THRESH_WWFO_TIME	4	binary	Writer wait-for-open threshold setting in minutes.
D0	SQIB_THRESH_RIDLE_TEXCP	4	binary	Accumulated count of reader idle threshold exceptions for life of subsystem
D4	SQIB_THRESH_RWFO_TEXCP	4	binary	Accumulated count of reader wait-for-open threshold exceptions for life of subsystem
D8	SQIB_THRESH_RWAIT_TEXCP	4	binary	Accumulated count of reader wait threshold exceptions for life of subsystem
DC	SQIB_THRESH_WIDLE_TEXCP	4	binary	Accumulated count of writer idle threshold exceptions for life of subsystem
E0	SQIB_THRESH_WWFO_TEXCP	4	binary	Accumulated count of writer wait-for-open threshold exceptions for life of subsystem
E4	SQIB_THRESH_WWAIT_TEXCP	4	binary	Accumulated count of writer wait threshold exceptions for life of subsystem
E8	SQIB_THRESH_PIPE1_TEXCP	4	binary	Reserved
EC	SQIB_THRESH_PIPE2_TEXCP	4	binary	Reserved
F0	SQIB_THRESH_PIPE3_TEXCP	4	binary	Reserved
F4	SQIB_THRESH_CONN1_TEXCP	4	binary	Reserved
F8	SQIB_THRESH_CONN2_TEXCP	4	binary	Reserved
FC	SQIB_THRESH_CONN3_TEXCP	4	binary	Reserved
100	SQIB_DATA_ARRAY	variable	array	The type of array depends on the request. See “Query Services SQIB Output Arrays” on page 192.

Query Services SQIB Output Arrays

Each output array from the query service has a different format and different size entries, depending on the specific request. The following list of requests includes the format and size or points to a table where you can find that information:

- Job list request (SQIB_JOBNAME). The array consists of 8-character job names and the count of entries in the array is in SQIB_ENTRY_COUNT.
- Pipe list request (SQIB_PIPENAME). The array consists of 44-character pipe names. The count of entries in the array is in SQIB_ENTRY_COUNT.
- Job status request (SQIB_STATUS_LIST). See “SQIB_STATUS_List Array.”
- Pipe status request (SQIB_STATUS_LIST). See “SQIB_STATUS_List Array.”
- Threshold status request (SQIB_THRESH_LIST). See “SQIB_THRESH_LIST Array” on page 193.

SQIB_STATUS_List Array: Each entry in the job or pipe status output array is X'6C' long and is described in the following table. Hexadecimal offsets represent offsets into each entry.

Figure 67 (Page 1 of 2). Format of Each Job and Pipe Status Array Entry

Hex Offset	Field Name	Length (bytes)	Attribute	Description
0	SQIB_A_JOBNAME	8	char	Jobname
8	SQIB_A_PIPENAME	decimal 44	char	Pipename
34	SQIB_A_DDNAME	8	char	DDNAME
3C	SQIB_A_BLKSIZE	4	binary	Block size used for pipe
40	SQIB_A_LRECL	4	binary	LRECL used for pipe
44	SQIB_A_XFRS	4	binary	Blocks or records transferred
48	SQIB_A_WAIT_COUNT	4	binary	Waits on full (if writer), waits on empty (if reader)
4C	SQIB_A_PIPE_DEPTH	4	binary	Pipe depth used
50	SQIB_A_TIMES	8	char	Timestamp of last I/O to pipe
58	SQIB_A_ACCUM_WAIT	8	char	Accumulated wait time in TOD units
60	SQIB_A_TOKEN	4	binary	Internal use
64	SQIB_A_IO	1	char	R=RDR, W=WTR
65	SQIB_PIPE_ATTR	1	bit	Pipe attributes
65	SQIB_F_WFO	1		X'80' means job is waiting for partner to open
65	SQIB_F_WAIT	1		X'40' means job is waiting on full if writer and empty if reader
65	SQIB_F_IDLE	1		X'20' means job is idle
65	SQIB_F_TIME_EXCD	1		X'10' means job wait/idle/wait-for-open time has exceeded threshold setting

Figure 67 (Page 2 of 2). Format of Each Job and Pipe Status Array Entry

Hex Offset	Field Name	Length (bytes)	Attribute	Description
65	SQIB_F_RECFM	1		Bits=X'0C' **X'00'** RECFM is unknown **X'04'** RECFM=V **X'08'** RECFM=F **X'0C'** RECFM=U
65	SQIB_F_BLKED	1		X'02' means records are blocked
65	SQIB_F_ALLOC_DONE	1		X'01' means allocation completed
66	SQIB_PIPE_ATTR2	1	bit	Additional Pipe attributes
66	SQIB_F_OPEN_DONE	1		X'80' means open completed
66	SQIB_F_ONE_IO_DONE	1		X'40' means at least 1 I/O done
66	SQIB_YEAR_OVERFLOW	1	bit	Time overflow flag
67	SQIB_TIME_EXCD_YEAR	1		X'80' means wait, idle or wait-for-open time > 1 year, it is invalid
68	SQIB_WAIT_IDLE_WFO_T	4	binary	Wait, idle, or wait-for-open time in seconds

SQIB_THRESH_LIST Array: Each entry in the threshold job-specific status output array is X'40' long and is described in the following table. Hexadecimal offsets represent offsets into each entry.

Figure 68. Format of Each Threshold Status Array Entry

Hex Offset	Field Name	Length (bytes)	Attribute	Description
0	SQIB_THRESH_TYPE	1	bit	Type of exception: 1=Reader idle 2=Reader waiting 3=Reader waiting-for-open 4=Writer idle 5=Writer waiting 6=Writer waiting-for-open
4	SQIB_THRESH_JOBNAME	8	char	Name of job that is a threshold exception
C	SQIB_THRESH_RW	1	char	Reader=R, Writer=W
10	SQIB_THRESH_TIME	4	binary	Time that the job has been a threshold exception, in seconds
14	SQIB_THRESH_PIPENAME	44 decimal	char	Pipename associated with the job that is threshold exception.

Example of a Job Status Query Request

The following assembler program requests job status. It obtains an XMRB, initializes the appropriate fields, and calls ASFPMSQI to learn the size of the SQIB. Once ASFPMSQI returns, the XMRB and SQIB are updated before the second call to ASFPMSQI to request job status for all active BatchPipes jobs in subsystem BP01. Note that the example does not perform any error-checking and is meant only to illustrate the use of the query service.

```
***********************************************************************
* Establish addressability
***********************************************************************
SAMPPRG  CSECT
SAMPPRG  AMODE 31
SAMPPRG  RMODE 24
         BALR  12,0                        Establish addressability
         USING *,12                        indicate base register
         STM   14,12,12(13)                standard linkage
         ST    13,MYSAVE+4
         LA    13,MYSAVE
***********************************************************************
*
* Load ASFPMSQI for later call
*
***********************************************************************
         LOAD  EP=ASFPMSQI,LOADPT=ASFPMSQI load ASFPMSQI for later call

***********************************************************************
*
* Obtain the storage for an XMRB
*
***********************************************************************
         STORAGE OBTAIN,LENGTH=LENXMRB,    obtain storage for XMRB     *
               ADDR=XMRBPTR,LOC=(ANY,ANY)
         L     10,XMRBPTR                  Set basing for XMRB
         USING XMRB,10                     Indicate basing for XMRB
         LA    2,XMRB_LEN                  base length of XMRB
         LA    2,XMRB_JOBNAME_LEN(,2)      Add length of one jobname
* Clear XMRB header plus length of one jobname
         MVI   XMRB,X'00'                  Clear XMRB for one jobname
         MVC   XMRB+1(XMRB_LEN+XMRB_JOBNAME_LEN-1),XMRB
         MVC   XMRBEYE,=C'XMRB'            Put eye catcher in XMRB
         MVI   XMRBVER,XMRBKVER            Indicate version number
         MVC   XMRB_SSNAME_REQ,SSNAME      Indicate Target Subsystem
         L     2,LENXMRB                   Get length of XMRB
         STCM  2,7,XMRB_SIZE               Save length into XMRB
         LA    2,XMRB_BUFF_SIZE            Get SQIB size request
         ST    2,XMRB_REQUEST              Indicate Request in XMRB

***********************************************************************
*
* Call ASFPMSQI to get the length needed for a SQIB
*
***********************************************************************
         L     15,ASFPMSQI                 Get address of ASFPMSQI
         CALL  (15),(XMRBPTR)              Call ASFPMSQI
```

```
***********************************************************************
*
* Store the size of the SQIB to obtain
*
***********************************************************************
         L     2,XMRB_SQIB_BUFF_SIZE       load SQIB size
         ST    2,LENSQIB                   Save SQIB size

***********************************************************************
*
* Obtain the storage for a SQIB
*
***********************************************************************
         STORAGE OBTAIN,LENGTH=LENSQIB,    Get a SQIB                  *
               ADDR=SQIBPTR,LOC=(ANY,ANY)
         L     9,SQIBPTR                   set base for SQIB
         USING SQIB,9                      Indicate basing for SQIB
         AR    2,9                         Calculate ending address
         ST    2,ENDSQIB                   Save for clearing SQIB
         MVI   SQIB,X'00'                  Clear head of SQIB
         MVC   SQIB+1(256),SQIB
         MVC   SQIBEYE,=C'SQIB'            Put eye catcher in SQIB
         MVI   SQIB_VER,SQIBKVER           Indicate version number
         L     2,LENSQIB                   Get size of SQIB
         STCM  2,7,SQIB_SIZE               Save size into SQIB
         ST    9,XMRB_SQIB_PTR             Chain the SQIB to XMRB
         ST    10,SQIB_XMRB_PTR            Chain the XMRB to SQIB

***********************************************************************
*
* Fill in the XMRB to indicate a JOBS STATUS request for all jobs
*
***********************************************************************
         LA    2,XMRB_JOBS_STATUS          Get request type
         ST    2,XMRB_REQUEST              Save request type in XMRB
         LA    2,1                         Indicate 1 entry in array
         ST    2,XMRB_ENTRY_COUNT          Update entry count
         LA    2,L'JOBNAME                 Length of jobname
         ST    2,XMRB_ENTRY_LENGTH         Update entry length
         MVC   XMRB_JOBNAME_ARRAY,JOBNAME    ' Put in '*' jobname
         DS    0H
***********************************************************************
*
* Call ASFPMSQI to get Job status
*
***********************************************************************
         L     15,ASFPMSQI                 get address of ASFPMSQI
         CALL  (15),(XMRBPTR)              Call ASFPMSQI

***********************************************************************
*
* Store return and error code, error handling not implemented in
* this program
*
***********************************************************************
         ST    15,RETCODE                  save return code
         MVC   ERRCODE,SQIB_ERR_CODE       save SQIB error code
```

```
***********************************************************************
*
* Clean up the XMRB and SQIB storage obtained
*
***********************************************************************
CLEANUP  DS 0H
         STORAGE RELEASE,LENGTH=LENSQIB,  free the SQIB                *
               ADDR=SQIBPTR
         STORAGE RELEASE,LENGTH=LENXMRB,  free the XMRB                *
               ADDR=XMRBPTR
* Delete ASFPMSQI after final use
         DELETE EP=ASFPMSQI                delete ASFPMSQI
         L     13,MYSAVE+4                 restore previous SA ptr
         LM    14,12,12(13)                restore callers registers
         BR    14                          return to caller

***********************************************************************
*
* Definitions
*
***********************************************************************
MYSAVE   DS    20F                         standard save area
LENXMRB  DC    F'2040'                     XMRB length - job status
*                                          calculated by multiplying
*                                          250 times length of
*                                          XMRB_JOBNAME (8) and adding
*                                          the length of XMRBHEAD (40)
*
LENSQIB  DS    F                           length of SQIB
XMRBPTR  DS    F                           Pointer to XMRB
SQIBPTR  DS    F                           Pointer to SQIB
RETCODE  DS    F                           Return code
ERRCODE  DS    F                           Error code
ENDSQIB  DS    F                           Pointer to end of SQIB
ASFPMSQI DS    A                           Address of ASFPMSQI (loaded)
SSNAME   DC    CL4'BP01'                   name of subsystem
JOBNAME  DC    CL8'*'                      wildcard jobname
***********************************************************************
*
* XMRB
*
***********************************************************************
         ASFPZXMR DSECT=YES
***********************************************************************
*
* SQIB
*
***********************************************************************
         ASFPZSQI DSECT=YES
         END
```

Appendix C: BatchPipes Version Information

This chapter describes General-use Programming Interfaces and associated guidance information.

BatchPipes provides a programming interface that allows you to verify the version of the BatchPipes code and determine if a data set defined by a specific DD statement is being managed by a BatchPipes subsystem.

IBM recommends that you do not write applications that depend upon a specific name to indicate whether the subsystem is BatchPipes or JES2 or JES3 or some other subsystem. Rather, **IBM recommends that you use the following two-step process**:

- Step 1: Find out if the data set is being managed by a subsystem
- Step 2: If the answer to Step 1 is YES, find out whether the subsystem is a BatchPipes subsystem.

Product-sensitive Programming Interface

Step 1: Is the data set managed by a subsystem? Determine from the data set's job file control block (JFCB) whether it has been allocated through a subsystem. Use the MVS/DFP* RDJFCB macro to obtain a copy of the data set's JFCB, then test the bit labeled JFCSDS in the field labeled JFCBTSDM. If that bit is set to B'1', the data set is managed by a subsystem — possibly BatchPipes or JES2 or JES3.

End of Product-sensitive Programming Interface

Step 2: Is the subsystem a BatchPipes subsystem? Use the MVS/ESA GETDSAB macro to obtain a pointer to the data set's data set allocation block (DSAB). The field named DSABSSNM contains the name of the subsystem. Specify that name on the subsystem interface (SSI) call type 54 (SSI 54) to obtain the subsystem version information (SSVI) for the referenced data set.

Do not compare fields SSVIVERS or SSVIFMID without first identifying field SSVICNAM. Fields SSVIFMID and SSVIVERS would be expected to change (somewhat randomly, perhaps) while SSVICNAM will remain constant.

Figure 69 describes the fields that BatchPipes sets in the SSOB and its extension, the SSVI, and the contents of those fields.

Figure 69 (Page 1 of 2). Fields BatchPipes Sets in the SSOB and the SSVI

Field name	Contents
SSVIVERS	CL8' 1.1.0' (right justified)
SSVIFMID	CL8'HTMT100 ' (left justified)
SSVICNAM	CL8'BP/MVS ' (left justified)
SSVIUDOF	X'0'
SSVIFLEN	SSVIFSIZ
SSVIRLEN	SSVIFSIZ + X'0027'

Figure 69 (Page 2 of 2). Fields BatchPipes Sets in the SSOB and the SSVI

Field name	Contents
SSVIVLEN	X'0027' followed by a character string: • C',BATCHPIPES_COMMAND_PREFIX=' • An apostrophe (') • BatchPipes command prefix • An apostrophe (') For example, a subsystem named BP01 might have the following string: `,BATCHPIPES_COMMAND_PREFIX='BP01    '`
SSOBRETN	One of the following: • SSVIOK (0) — successful completion (the caller provided a large enough data area, in a correct storage key, and initialized necessary fields, and so forth.) • SSVINSTR (8) — the caller did not pass enough storage to SSI 54 to contain the entire response. However, if at least SSVIFLEN bytes were passed, BatchPipes returns the fixed portion of the response (not including SSVIVLEN or SSVIDAT) with SSVINSTR value • SSVIPARM (16) — the caller did not initialize one or more necessary fields accurately (that is, SSVIID, SSVIVER, SSVILEN or SSOBINDV.) In this case, the contents of the remaining response area are unpredictable and should be ignored. • SSVIABLG (24) — a serious error (possibly an ABEND) was intercepted during execution. In this case, the contents of the remaining response area are unpredictable and should be ignored.

Example of Using the Interface: Find out whether the data set defined by the following DD statement is to be managed by a BatchPipes subsystem, and if so, which BatchPipes subsystem:

```
//DDIN  DD  DSN=MY.PIPED.DATA,SUBSYS=BP01,RECFM=FB,LRECL=80,DISP=SHR
```

Given the preceding DD statement of a program that follows the data chain to find out whether a given DD statement is referencing a BatchPipes subsystem, and if so, which of several BatchPipes subsystems is referenced (for example, Production BatchPipes is BP01 and Test BatchPipes is BT01). The same program also finds the value of the command prefix for the JCL-referenced subsystem.

```
* Here is the sample code.
SSI54TX  CSECT ,
SSI54TX  AMODE 31
SSI54TX  RMODE 24
         STM   14,12,12(13)      SAVE CALLERS REGISTERS
         BALR  12,0              ABSOLUTE ADDRESSABILITY
         USING *,12              INFORM ASSEMBLER OF SAME
         ST    13,MYSAVE+4
         LA    13,MYSAVE         (NON-REENTRANT EXAMPLE)

*************************************************
*  Step 1: Is the data set managed by a subsystem?
*************************************************
         RDJFCB (MYDCB,INPUT)        GET JFCB FOR INPUT DATASET
*
         TM    JFCBTSDM,JFCSDS       CHECK IF A SUBSYSTEM DATASET
         BNO   NOT@SDS               IF NOT SDS THEN SKIP SDS PROCESS
```

```
*************************************************

*  Step 2: Is the subsystem a BatchPipes subsystem?
*************************************************
         GETDSAB DCBPTR=MYDCBPTR,DSABPTR=DSABPTR
         LTR   15,15                 CHECK FOR SUCCESSFUL RC
         BZ    GOT@DSAB
         WTO   'READER FAILED TO GET DSAB'
         B     OPENFAIL
GOT@DSAB DS    0H
         WTO   'READER FOUND DSAB FOR SUBSYSTEM DATASET'
         LA    6,MYSSVI
         USING SSVI,6
         L     5,DSABPTR
         USING DSAB,5
         LA    4,MYSSOB
         USING SSOB,4
         LA    3,MYSSIB
         USING SSIB,3

* FILL IN THE SSOB
         MVC   SSOBID,=C'SSOB'
         LA    8,SSOBHSIZ
         STH   8,SSOBLEN
         ST    3,SSOBSSIB
         ST    6,SSOBINDV
* FILL IN THE SSIB
         MVC   SSIBID,=C'SSIB'
         LA    8,SSIBSIZE
         STH   8,SSIBLEN
         MVC   SSIBSSNM,DSABSSNM   GET SSNAME FROM DSAB
* FILL IN THE SSVI
         MVC   SSVIID,=A(SSVICID)
         L     8,=A(L'MYSSVI)    ARBITRARY SIZE FOR 1ST GUESS
         STH   8,SSVILEN
         MVI   SSVIVER,SSVICVER

* NORMALLY THE VALIDITY OF THE SSNAME WOULD BE CHECKED BEFORE
* ACTUALLY CALLING THE SUBSYSTEM, BUT SINCE THIS SS RESPONDED
* TO AN ALLOCATION REQUEST, ASSUME IT IS OPERATIONAL.
         MVC   SSOBFUNC,=Y(SSOBSSVI)
         OI    SSPARM,X'80'
         MVC   SSIBSSNM,DSABSSNM   USE SSNAME FROM DSAB
         LA    1,SSPARM
         IEFSSREQ ,              ISSUE SSI REQUEST
         LTR   8,15              KEEP SSREQ RC SAFE IN REGISTER 8
         BZ    SSREQ OK
         WTO   'NONZERO RC FROM IEFSSREQ',ROUTCDE=11
         C     8,=A(SSRTNSUP)    DOES SUBSYSTEM SUPPORT SSI 54?
         BNE   SSRQNSUP           THAT ISN'T THE PROBLEM
         WTO   'DDIN SUBSYS DOES NOT SUPPORT SSI 54',ROUTCDE=11
* THIS IS TRUE OF JES2 AND MSTR SUBSYSTEMS PRIOR TO MVS/ESA 4.3 ...
* ...AND IS TRUE OF BATCHPIPES/MVS PRIOR TO PTF UN59629.
         B     OPENFAIL
```

```
SSRQNSUP DS     0H
         ABEND 5,DUMP            REGISTER 8 HAS IEFSSREQ RETURN CODE
SSREQ OK DS     0H
         ICM   2,15,SSOBRETN     SEE IF SSVI CALL WAS SUCCESSFUL
         BZ    SSOB@OK            YES - SEE IF CL8'BP/MVS ' SUBSYS
         C     2,=A(SSVINSTR)     NO - WAS 256 BYTES TOO SMALL ?
         BE    SSOB@OK            YES - FIXED HEADER FILLED IN  ANYWAY?
* NOTE: WHETHER A GIVEN SUBSYSTEM CONSIDERS SSOBRETN=SSVINSTR TO BE...
*       ...A FATAL USER ERROR OR NOT SEEMS TO BE UP TO THE SUBSYSTEM.
*       IT IS REASONABLE TO RETURN SOME MINIMUM AMOUNT OF DATA WHICH...
*       ...FITS INTO THE SUPPLIED AREA (AS BATCHPIPES DOES).
*       IT IS ALSO REASONABLE TO RETURN NO DATA UNLESS IT ALL FITS...
*       ...(AS SOME OTHER SUBSYSTEMS MIGHT DO).
         ABEND 2,DUMP            ELSE WHAT DO SSVI / SSOB LOOK LIKE?
*        WTO   'NONZERO RC IN SSOB'
SSOB@OK  DS    0H                SSI 54 (SSVI) CALL SUCCESSFUL
         CLC   SSVICNAM,PIPECNAM  IS COMMON NAME AS HOPED?
         BNE   NOT@PIPES           NO - NOT BATCHPIPES SUBSYS

* IF WE GET HERE, THEN IT IS A BATCHPIPES/MVS DATASET
         WTO   'DDIN IS A BATCHPIPES/MVS DATASET',ROUTCDE=11
         MVC   CNAMWTO+9(L'SSVICNAM),SSVICNAM  CL8'BP/MVS  '
         WTO   MF=(E,CNAMWTO)
         MVC   CNAMWTO+9(L'SSVIVERS),SSVIVERS  CL8'   1.1.0'  PROBABLY
         MVC   CNAMWTO+4(4),=C'VERS'
         WTO   MF=(E,CNAMWTO)
         MVC   CNAMWTO+9(L'SSVIFMID),SSVIFMID  CL8'HTMT100 '  PROBABLY
         MVC   CNAMWTO+4(4),=C'FMID'
         WTO   MF=(E,CNAMWTO)
         ABEND 1,DUMP            LETS SEE WHAT BP/MVS SSVI LOOKS LIKE
CNAMWTO  WTO   'CNAM=         ',ROUTCDE=11,DESC=7,MF=L
NOT@PIPES DS    0H
         WTO   'DDIN IS NOT A BATCHPIPES/MVS DATASET',ROUTCDE=11
         CLC   SSVICNAM,JES2CNAM  PERHAPS A JES2 SUBSYS DS ?
         BNE   NOT@JES2            NO - RUNNING OUT OF GUESSES
         WTO   'DDIN IS A JES2 DATASET'
         ABEND 4,DUMP            LET'S SEE WHAT THE SSVI LOOKS LIKE
```

```
NOT@JES2  DS   0H
         CLC   SSVICNAM,JES3CNAM  PERHAPS A JES3 SUBSYS DS ?
         BNE   NOT@JES             NO - RUNNING OUT OF GUESSES
         WTO   'DDIN IS A JES3 DATASET'
         ABEND 4,DUMP            LET'S SEE WHAT THE SSVI LOOKS LIKE
NOT@JES  DS    0H
         WTO   'DDIN IS NOT A JES2 OR JES3 DATASET'
         CLC   SSVICNAM,MSTRCNAM  PERHAPS A MSTR SUBSYS DS ?
         BNE   NOT@MSTR            NO - RUNNING OUT OF GUESSES
         WTO   'DDIN IS A MSTR DATASET'  HOW DID THIS HAPPEN ?
         ABEND 4,DUMP            LET'S SEE WHAT THE SSVI LOOKS LIKE
NOT@MSTR DS    0H
         WTO   'DDIN IS SUPPORTED BY AN UNRECOGNIZABLE SUBSYS'
* THIS COULD BE BATCH LSR (IBM) OR SOME OEM OR USER-WRITTEN SUBSYSTEM.
*  (AS OF THIS WRITING, BATCH LSR DOES NOT SUPPORT SSI 54 CALLS.)
         ABEND 3,DUMP            LET'S SEE WHAT THE SSVI LOOKS LIKE
NOT@SDS  DS    0H               DDIN(MYDCB) POINTS TO NON-SUBSYSTEM
         OPEN  (MYDCB,INPUT),TYPE=J  RDJFCB WAS DONE, USE IT FOR OPEN
         LTR   15,15
         BNZ   OPNFAILD         (INSUFFICIENT TEST FOR OPEN SUCCESS)
         TM    MYDCB+(DCBOFLGS-IHADCB),DCBOFOPN  WAS OPEN REALLY OKAY?
         BO    OPENOKAY           YES, OPEN WAS REALLY SUCCESSFUL

OPNFAILD DS    0H
         WTO   'OPEN FAILED',ROUTCDE=11
         B     OPENFAIL
OPENOKAY DS    0H
*              . . . FILE I/O PROCESSING GOES HERE . . .
EOF      DS    0H
         WTO   'READER CLOSING',ROUTCDE=11
         CLOSE (MYDCB)
OPENFAIL DS    0H
         WTO   'READER ENDING',ROUTCDE=11
         L     13,MYSAVE+4       LOAD POINTER TO CALLERS SAVE AREA
         L     14,12(13)         LOAD RETURN REGISTER
         LM    0,12,20(13)       RESTORE MOST OF CALLERS REGISTERS
         SLR   15,15             COND CODE 0000
         BR    14                (RETURN TO CALLER W/ RC=0000 SET)
MYSAVE   DS    20F               NON-REENTRANT SAVE AREA FOR EXAMPLE
DSABPTR  DS    F
SSPARM   DC    A(MYSSOB)
MYDCBPTR DC    A(MYDCB)
```

```
*
PIPECNAM DC    CL8'BP/MVS  ' <---THIS IS BATCHPIPES' COMMON NAME
JES2CNAM DC    CL8'JES2    ' <---THIS IS JES2'S COMMON NAME
MSTRCNAM DC    CL8'MASTER  ' <---THIS IS MSTR SUBSYS FOR MVS/ESA 4.3
JES3CNAM DC    CL8'JES3    ' <---UNINSPIRED GUESS AT JES3'S COMMON NAME
*
MYDCB    DCB   RECFM=FB,LRECL=80,DDNAME=DDIN,
               DSORG=PS,MACRF=GL,EODAD=EOF,EXLST=LSTA
LSTA     DC    0F'0',X'87',AL3(JFCBAREA)  RDJFCB EXIT LIST
JFCBAREA DS    0F
MYJFCB   IEFJFCBN LIST=NO
         DC    C'MYSSOB'         EYECATCHER
MYSSOB   DC    (SSOBHSIZ)X'00'
         DC    C'MYSSIB'         EYECATCHER
MYSSIB   DC    (SSIBSIZE)X'00'
         DC    C'MYSSVI'         EYECATCHER
MYSSVI   DC    256X'00'          LARGE AREA FOR EXAMPLE SSREQ TYPE 54
IOAREA   DSECT
IOBUF    DS    CL80              INPUT RECORD AREA
         IEFJSSIB ,              SSIB
         IEFSSOBH ,              SSOB
SSOBGN   EQU   *                        CONNECT SSVI W/ SSOB
         IEFSSVI ,             SSVI MACRO 1ST SHIPPED WITH MVS/ESA 4.3
         IHADSAB ,               DSAB
         CVT DSECT=YES,LIST=NO   CVT
         IEFJESCT ,              SSCT
         DCBD  DSORG=PS,DEVD=DA  IHADCB MAPPING MACRO
         END   ,
```

For more information, see *MVS/ESA Using the Subsystem Interface*.

Appendix D: BatchPipes Messages

ASFP000I BATCHPIPES FOR OS/390 SUBSYSTEM *ssname*:
PRODID=*productid* PRODLVL=*productlvl* COMPID=*compid*
CONTAINS LICENSED MATERIALS - PROPERTY OF IBM CORP.
CONTAINS RESTRICTED MATERIALS OF IBM CORP.
5655-D45 (C) COPYRIGHT IBM CORP. 1992, 2000
ALL RIGHTS RESERVED.
U.S. GOVERNMENT USERS RESTRICTED RIGHTS -
USE, DUPLICATION, OR DISCLOSURE RESTRICTED BY
GSA ADP SCHEDULE CONTRACT WITH IBM CORP.

Explanation: This is a proprietary statement concerning the use of the BatchPipes product. The second line of the message indicates the product id, product level, and component id of the product.

In the message text:

ssname
is the name of the BatchPipes subsystem.

productid
is the BatchPipes product id.

productlvl
is the BatchPipes product level.

compid
is the BatchPipes component id.

Source: BATCHPIPES

Detecting Module:
ASFPMISP

System Action: BatchPipes subsystem processing continues

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP001E BATCHPIPES COMPONENT NOT STARTED - NOT IN AN APF AUTHORIZED LIBRARY

Explanation: BatchPipes detected that it was not installed in an APF authorized library.

Source: BATCHPIPES

Detecting Module:
ASFPMIKE

System Action: The BatchPipes subsystem address space is ended.

Operator Response: Notify your system programmer.

User Response: None.

Application Programmer Response: None.

System Programmer Response: Ensure that the BatchPipes product is installed in an APF authorized library.

ASFP005I BATCHPIPES SUBSYSTEM JOB NAME *jobname* IS NOT VALID.

Explanation: The BatchPipes procedure name, which is used as the name of the subsystem, is not a valid subsystem name. It must be a 1-4 character name.

In the message text:

jobname
is the job name for the BatchPipes started procedure.

Source: BATCHPIPES

Detecting Module:
ASFPMISP

System Action: The BatchPipes address space is ended.

Operator Response: Notify your system programmer.

User Response: None.

Application Programmer Response: None.

System Programmer Response: Ensure that the BatchPipes procedure name is a valid subsystem name (1-4 characters).

ASFP006I BATCHPIPES *ssname* INPUT COMMAND PREFIX (*badprefx*) IS NOT VALID. DEFAULT PREFIX (*defprefx*) USED.

Explanation: An command prefix value *badprefx* that was specified as input to the BatchPipes subsystem *ssname* is not valid. The prefix value is either greater than 8 characters, begins with an invalid symbol, or contains an imbedded blank.

In the message text:

ssname
is the name of the BatchPipes subsystem.

badprefx
is the invalid command prefix value that was specified.

defprefx
is the default value to be used as the command profix for the subsystem.

Source: BATCHPIPES

Detecting Module:
ASFPMISP

System Action: BatchPipes subsystem initialization continues. The specified command prefix value is ignored and as a default, the subsystem name *ssname* is used as the command prefix.

Operator Response: If the default command prefix is unacceptable, enter a CANCEL command prefixed by the *ssname* to cancel the BatchPipes subsystem immediately. Then if the bad command prefix was entered by you as a parameter on the START command when you started BatchPipes, re-enter the START command to restart the BatchPipes subsystem with a valid command prefix value.

If the bad command prefix was not specified by you on the START command, notify your system programmer.

User Response: None.

Application Programmer Response: None.

System Programmer Response: If the bad command prefix was specified as a parameter in the job procedure that was used to start the BatchPipes subsystem, correct the command prefix parameter in the procedure so the next time the subsystem is started, the desired command prefix will be used.

ASFP007I BATCHPIPES *ssname* COMMAND PREFIX IS *cmdprefix*

Explanation: The command prefix that is being used by the BatchPipes subsystem *ssname* is *cmdprefix*. All commands entered for the subsystem should begin with this prefix.

In the message text:

ssname
is the name of the BatchPipes subsystem.

cmdprefix
is a 1-8 character command prefix.

Source: BATCHPIPES

Detecting Module:
ASFPMISP

System Action: BatchPipes subsystem initialization continues.

Operator Response: Whenever you enter one of the commands supported by the BatchPipes subsystem, you must begin the command with the indicated command prefix (e.g., *cmdprefix* STATUS).

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP008I BATCHPIPES *ssname* MEM= PARAMETER VALUE IS NOT VALID. IT IS IGNORED.

Explanation: The MEM= parameter value that was specified on the START command or in the started procedure is not valid. Either the member suffix that was specified is greater than two characters or contains a non-alphanumeric character.

In the message text:

ssname
is the name of the BatchPipes subsystem.

Source: BATCHPIPES

Detecting Module:
ASFPMISP

System Action: BatchPipes subsystem initialization continues. If the BatchPipes default PARMLIB member, ASFPBP00, exists, the subsystem will attempt to use that member. If it does not exist, the subsystem will assign default values to all BatchPipes parameters that can be specified in a PARMLIB member.

Operator Response: If the bad MEM= parameter value was entered on the START command, wait for the BatchPipes subsystem to become active. Then use the BatchPipes "SET,MEM=xx" command to force the subsystem to use the PARMLIB member that you had intended to specify. Otherwise, notify your system programmer.

User Response: None.

Application Programmer Response: None.

System Programmer Response: If the bad MEM= parameter value was specified as a parameter in the job procedure that was used to start the BatchPipes subsystem, wait for the BatchPipes subsystem to become active. Then use the BatchPipes "SET,MEM=xx" command to force the subsystem to use the PARMLIB member that you had intended to specify. Also, correct the parameter value in the job procedure so the next time the subsystem is started, the desired PARMLIB member will be used.

ASFP009E BATCHPIPES *ssname* SUBSYSTEM MUST BE A STARTED TASK.

Explanation: The BatchPipes subsystem *ssname* must be a started task.

In the message text:

ssname
is the name of the BatchPipes subsystem.

Source: BATCHPIPES

Detecting Module:
ASFPMISP

System Action: The BatchPipes address space is terminated.

Operator Response: Start BatchPipes as a started task.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP011I BATCHPIPES *ssname* INITIALIZATION COMPLETE.

Explanation: The BatchPipes subsystem *ssname* has completed its initialization.

In the message text:

ssname
is the name of the BatchPipes subsystem.

Source: BATCHPIPES

Detecting Module:
ASFPMISP

System Action: The BatchPipes subsystem continues processing. It is now ready to process any data set whose DD statement specifies the SUBSYS= keyword indicating this subsystem (i.e., *ssname*).

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP012I BATCHPIPES *ssname* ENDED.

Explanation: The BatchPipes subsystem *ssname* has completed shutting itself down.

In the message text:

ssname
is the name of the BatchPipes subsystem.

Source: BATCHPIPES

Detecting Module:
ASFPMISP

System Action: The BatchPipes subsystem address space ends.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP013I BATCHPIPES *ssname* INITIALIZATION FAILED.

Explanation: The BatchPipes subsystem *ssname* could not be successfully initialized. The specific initialization error is indicated by a BatchPipes error message or abend that was issued just before this message.

In the message text:

ssname
is the name of the BatchPipes subsystem.

Source: BATCHPIPES

Detecting Module:
ASFPMISP

System Action: The BatchPipes subsystem ends.

Operator Response: Notify your system programmer.

User Response: None.

Application Programmer Response: None.

System Programmer Response: Examine the SYSLOG to determine the error message or abend that caused initialization to fail and then correct the condition that caused the error.

ASFP014I BATCHPIPES *ssname* ALREADY ACTIVE.

Explanation: One instance of the BatchPipes subsystem *ssname* already exists. Starting more than one BatchPipes subsystem with the same name is not allowed.

In the message text:

ssname
is the name of the BatchPipes subsystem.

Source: BATCHPIPES

Detecting Module:
ASFPMISP

System Action: The BatchPipes subsystem that issued this message ends.

Operator Response: Ensure you specified the correct name on the START command when starting the BatchPipes subsystem.

If you were restarting the BatchPipes subsystem after an error situation, ensure that the first instance of the subsystem ends before issuing the START command.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP016I BATCHPIPES *ssname* IS NOT A VALID SUBSYSTEM.

Explanation: The BatchPipes subsystem *ssname* is not a defined subsystem name known by the system.

In the message text:

ssname
is the name of the BatchPipes subsystem.

Source: BATCHPIPES

Detecting Module:
ASFPMISP

System Action: The BatchPipes subsystem ends.

Operator Response: Notify your system programmer.

User Response: None.

Application Programmer Response: None.

System Programmer Response: Ensure the BatchPipes subsystem name *ssname* is defined as a valid subsystem name in the IEFSSNxx PARMLIB member that was used to IPL the system.

ASFP017I BATCHPIPES *ssname* MODE(*mode*) PIPEPLEX(*pipeplex-name*)

Explanation: This message is an informational message indicating the processing mode of the BatchPipes subsystem *ssname* and the pipeplex the subsystem is a part of, if any.

In the message text:

ssname
is the name of the BatchPipes subsystem.

mode
One of the following:

LOCAL
indicates the subsystem is running in local mode. In this mode, all pipes supported by the subsystem can only be used to pipe data between jobs executing on the same system as the subsystem.

XSYS
indicates the subsystem is running in cross-system mode. In this mode, all pipes supported by the subsystem can be used to pipe data between jobs executing on different systems which are part of the same pipeplex as this system.

pipeplex-name
is the name of the pipeplex the subsystem is a member of. **NONE** is indicated if the subsystem is not a member of any pipeplex such as is the case when the subsystem is running in LOCAL mode.

Source: BATCHPIPES

Detecting Module:
ASFPMISP

System Action: The BatchPipes subsystem continues processing.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP018I BATCHPIPES *ssname* COUPLING FACILITY STRUCTURE ERROR: INSUFFICIENT SPACE ALLOCATED FOR *strname*. POLICY-SIZE=*policysize* ALLOCATED-SIZE=*strsize*

Explanation: The BatchPipes subsystem *ssname* when initializing for cross-system piping was unable to allocate sufficient space on a Coupling Facility (CF) for the structure it requires to do cross-system piping. The size allocated is not only less than what was specified in the Coupling Facility Resource Management (CFRM) policy, but is less than the minimum space required to perform cross-system piping.

In the message text:

ssname
is the name of the BatchPipes subsystem.

strname
is the name of the BatchPipes Coupling Facility Structure.

policysize
is the size of the BatchPipes Coupling Facility Structure that was specified in the CFRM Policy in units of 4K.

strsize
is the size of the BatchPipes Coupling Facility Structure that was actually allocated in units of 4K.

Source: BATCHPIPES

Detecting Module:
ASFPMXI

System Action: The BatchPipes subsystem ends.

Operator Response: Notify your system programmer.

User Response: None.

Application Programmer Response: None.

System Programmer Response: Update your coupling facility configuration and your CFRM policy to ensure that there is sufficient space available on a coupling facility for the BatchPipes Coupling Facility Structure and then restart the BatchPipes subsystem.

ASFP019I BATCHPIPES *ssname* COUPLING FACILITY STRUCTURE ERROR: INSUFFICIENT POLICY SIZE FOR *strname*. POLICY-SIZE=*policysize*

Explanation: The BatchPipes subsystem *ssname* when initializing for cross-system piping was determined that the size of the BatchPipes Coupling Facility (CF) structure specified in the Coupling Facility Resource Management (CFRM) policy is too small to perform cross-system piping.

In the message text:

ssname
is the name of the BatchPipes subsystem.

strname
is the name of the BatchPipes Coupling Facility Structure.

policysize
is the size of the BatchPipes Coupling Facility Structure that was specified in the CFRM Policy in units of 4K.

Source: BATCHPIPES

Detecting Module:
ASFPMXI

System Action: The BatchPipes subsystem ends.

Operator Response: Notify your system programmer.

User Response: None.

Application Programmer Response: None.

System Programmer Response: Recalculate the size of the BatchPipes Coupling Facility Structure. Then update the size of the structure in the CFRM policy and restart the BatchPipes subsystem.

ASFP020I BATCHPIPES *ssname* COUPLING FACILITY STRUCTURE WARNING: *strname* ALLOCATED SIZE IS LESS THAN REQUESTED. POLICY-SIZE=*policysize* ALLOCATED-SIZE=*strsize*

Explanation: This is a warning message to indicate that the size of the BatchPipes Coupling Facility (CF) Structure that was allocated for doing cross-system piping is less than the size that was actually requested in the CFRM policy for the structure. The BatchPipes subsystem can continue initialization and perform cross-system piping, but will not be able to support as many cross-system pipes.

In the message text:

ssname
is the name of the BatchPipes subsystem.

strname
is the name of the BatchPipes Coupling Facility Structure.

policysize
is the size of the BatchPipes Coupling Facility Structure that was specified in the CFRM Policy in units of 4K.

strsize
is the size of the BatchPipes Coupling Facility Structure that was actually allocated in units of 4K.

Source: BATCHPIPES

Detecting Module:
ASFPMXI

System Action: The BatchPipes subsystem processing continues.

Operator Response: Notify your system programmer.

User Response: None.

Application Programmer Response: None.

System Programmer Response: Update your coupling facility configuration and your CFRM policy to ensure that there is sufficient space available on a coupling facility for the BatchPipes Coupling Facility Structure the next time the BatchPipes subsystem is restarted.

ASFP021I BATCHPIPES *ssname* UNABLE TO JOIN PIPEPLEX *pipeplex-name*. A PRIOR CONNECTION TO THE PIPEPLEX FOR *ssname* ON SYSTEM *sysname* STILL EXISTS AND IS UNRECOVERABLE.

Explanation: The BatchPipes subsystem, *ssname*, when attempting to join the BatchPipes pipeplex, *pipeplex-name*, detected that a previous instance of the subsystem on system, *sysname*, had joined the pipeplex and was subsequently ended. BatchPipes recovery was unable to successfully cleanup the subsystem's connection to the pipeplex and thus marked the connection as unrecoverable. Because of this, data related to that instance of the subsystem on system, *sysname*, might remain in existence in the BatchPipes coupling facility structure and may potentially cause problems if a new instance of the same subsystem on the same system is allowed to join the pipeplex. As a result, the new instance of the BatchPipes subsystem, *ssname*, on system, *sysname*, is prevented from joining the BatchPipes pipeplex.

In the message text:

ssname
is the name of the BatchPipes subsystem.

pipeplex-name
is the name of the BatchPipes pipeplex.

sysname
is the name of the system on which the BatchPipes subsystem is initializing.

Source: BATCHPIPES

Detecting Module:
ASFPMXI

System Action: The BatchPipes subsystem ends.

Any job which specifies the BatchPipes subsystem on one of its DD statement and which has not yet been scheduled into execution can still be scheduled for execution and serviced by other BatchPipes subsystems that are members of the same pipeplex.

Operator Response: To start the given BatchPipes subsystem on the indicated system, the entire pipeplex will have to be quiesced. To quiesce the pipeplex, each BatchPipes subsystem that is a member of the indicated pipeplex must be either stopped or canceled. This should be done at a point in time that will have the least impact on your installation's production work.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP022I BATCHPIPES *ssname* UNABLE TO JOIN PIPEPLEX *pipeplex-name*. CURRENT LEVEL OF EXISTING PIPEPLEX STRUCTURE IS NOT SUPPORTED BY LEVEL OF SUBSYTEM BEING STARTED.

Explanation: The BatchPipes subsystem, *ssname*, when attempting to join the BatchPipes pipeplex, *pipeplex-name*, detected that the current level of the pipeplex control block structure on the coupling facility is not a level it supports.

In the message text:

ssname
is the name of the BatchPipes subsystem.

pipeplex-name
is the name of the BatchPipes pipeplex.

Source: BATCHPIPES

Detecting Module:
ASFPMXI

System Action: The BatchPipes subsystem that is being started is abnormally terminated.

Operator Response: Notify your system programmer.

User Response: None.

Application Programmer Response: None.

System Programmer Response: To run the level of BatchPipes subsystem that was being started, you must run it in a different pipeplex than the pipeplex that it was attempting to join. Since the subsystem name is used to associate a BatchPipes subsystem with a given pipeplex, ensure that the subsystem PROC that was started specifies the correct level of BatchPipes libraries. If not, correct the library specification. Otherwise, if the level of BatchPipes you are trying to run is correct, then you need to establish a different pipeplex for that level of BatchPipes. Refer to the BatchPipes/MVS Users Guide and Reference for information about how to define the pipeplex.

ASFP023I BATCHPIPES *ssname* UNABLE TO JOIN PIPEPLEX *pipeplex-name*. A REBUILD OF THE CF STRUCTURE IS IN PROGRESS.

Explanation: The BatchPipes subsystem, *ssname*, when attempting to join the BatchPipes pipeplex, *pipeplex-name*, detected that a rebuild of the BatchPipes coupling facility (CF) structure is currently in progress and as a result, BatchPipes initialization is unable to continue. Either the BatchPipes subsystem is not yet at a point in initialization where it can participate in the rebuild process, or the rebuild has progressed to a point where new members are not allowed to connect to the structure at this time.

In the message text:

ssname
is the name of the BatchPipes subsystem.

pipeplex-name
is the name of the BatchPipes pipeplex.

Source: BATCHPIPES

Detecting Module:
ASFPMXI

System Action: The BatchPipes subsystem that is being started is abnormally terminated.

Operator Response: Wait for the rebuild of the coupling facility structure to complete and then restart the BatchPipes subsystem.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP024I BATCHPIPES *ssname* **ON SYSTEM** *sysname* **IS WAITING TO JOIN PIPEPLEX** *pipeplex-name***.**

Explanation: The BatchPipes subsystem, *ssname*, on system, *sysname*, is waiting to join the BatchPipes pipeplex, *pipeplex-name*. The subsystem is not immediately able to join the pipeplex because a prior member of the pipeplex with either the same system name or same coupling facility connection ID still needs to be cleaned up by BatchPipes subsystem/system termination processing on one of the other active systems in the pipeplex.

In the message text:

ssname
is the name of the BatchPipes subsystem.

sysname
is the name of the system on which the BatchPipes subsystem is initializing.

pipeplex-name
is the name of the BatchPipes pipeplex.

Source: BATCHPIPES

Detecting Module:
ASFPMXI

System Action: BatchPipes subsystem initialization will wait a short interval and then try to join the pipeplex again.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP033E BATCHPIPES *ssname* **DIRECTED LOAD OF** *modname* **FAILED.**

Explanation: Module *modname* was not found during initialization of the BatchPipes subsystem.

In the message text:

ssname
is the name of the BatchPipes subsystem.

modname
is the name of a module.

Source: BATCHPIPES

Detecting Module:
ASFPMIDL

System Action: The BatchPipes subsystem ends.

Operator Response: Notify your system programmer.

User Response: None.

Application Programmer Response: None.

System Programmer Response: Ensure the module *modname* is correctly installed.

ASFP081A BATCHPIPES *ssname* **TASK** *taskname* **FAILED. SPECIFY ACTION** *action*

Explanation: Task *taskname* has ABENDed during its processing. Specify the action that is to be taken because of this failure.

The actual list of actions presented is based on the importance of the task to the processing of the BatchPipes subsystem. If the task is vital to the processing of the subsystem then the 'IGNORE' option is not presented, otherwise the complete list (TERMSS, RESTART, IGNORE) is presented.

In the message text:

ssname
is the name of the BatchPipes subsystem.

taskname
is the descriptive name of the task in the BatchPipes address space that has failed. The possible tasks are:

ACCOUNTING Generates the Pipes SMF Type 91 records.

COMMANDS Provides general command processing for the BatchPipes subsystem.

STOP/MODIFY Provides specific command support for the BatchPipes subsystem.

THRESHOLDS Provides the monitoring of the inactivity thresholds for the BatchPipes subsystem.

Task Error Provides the error processing control for failed tasks in the BatchPipes address space.

X-SYS MESSAGE Provides internal cross-system communications for the PipePlex.

XSYS MSG PROCESS Executes internal cross-system communications services for the PipePlex.

X-SYS SERVICE Provides the services that are used to access the coupling facility dataset.

X-SYS TERM Provides the services that are used to clean-up for a failed system in the pipeplex. '

action
is the list of possible actions that can be requested. The possible actions are:

TERMSS Requests that the BatchPipes subsystem be cancelled.

RESTART Requests that the failing task be reinitialized.

IGNORE Requests that the task failure be bypassed.

Source: BATCHPIPES

Detecting Module:
ASFPMITM

System Action: The BatchPipes task manager function waits on the reply to this request for an action for 5 minutes. If no action is specified in the allotted time, then a default action is taken, the default action is 'IGNORE' (if it is a valid option for the task) or 'TERMSS'.

Operator Response: Notify your system programmer.

User Response: None.

Application Programmer Response: None.

System Programmer Response: Determine the importance of the *taskname* to the installation and provide the appropriate action.

The actual task failure can be determined by searching the SYSLOG, LOGREC data or other diagnostics for the system.

ASFP082I BATCHPIPES *ssname* **TASK** *taskname* **FAILED. NO RESPONSE TO ASFP081A RECEIVED. DEFAULT ACTION BEING TAKEN**

Explanation: Task *taskname* has ABENDed during its processing and Action message ASFP081A was displayed on the console but no action was specified after the allotted time (5 minutes). The result is that the default action of the specified task is being taken.

In the message text:

ssname
is the name of the BatchPipes subsystem.

taskname
is the descriptive name of the task in the BatchPipes address space that has failed.

Source: BATCHPIPES

Detecting Module:
ASFPMITM

System Action: The BatchPipes subsystem takes the default action for the failing task *taskname*. The default action is either: 'IGNORE' in which case message ASFP083I will be displayed for the task or 'TERMSS' which results in message ASFP218I and the CANCEL of the subsystem.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP083I BATCHPIPES *ssname* **TASK** *taskname* **FAILED.**

Explanation: Task *taskname* has ABENDed during its processing and the response to message ASFP081A was 'IGNORE'.

In the message text:

ssname
is the name of the BatchPipes subsystem.

taskname
is the descriptive name of the task in the BatchPipes address space that has failed.

Source: BATCHPIPES

Detecting Module:
ASFPMITM

System Action: The failed task is detached and its processing has ended.

Operator Response: Notify your system programmer.

User Response: None.

Application Programmer Response: None.

System Programmer Response: The task failure can be determined by either searching the SYSLOG or LOGREC data for the system.

ASFP084I BATCHPIPES *ssname* **INVALID RESPONSE (***response***) TO MESSAGE ASFP081A.**

Explanation: Task *taskname* has ABENDed during its processing and the response to message ASFP081A was invalid.

In the message text:

ssname
is the name of the BatchPipes subsystem.

response
  is the invalid response to the ASFP081A message which was specified.

Source: BATCHPIPES

Detecting Module:
ASFPMITM

System Action: The WTOR message (ASFP081A) is re-displayed.

Operator Response: Provide a valid response to the message.

Check the documentation of message ASFP081A for the list of valid responses.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP201I BATCHPIPES *ssname cmdname* **COMMAND ERROR --** *parmname* **PARAMETER VALUE IS TOO LONG.**

Explanation: The command entered, *cmdname*, specified a *parmname* value that is longer than the maximum length allowed for the given parameter.

In the message text:

ssname
  is the name of the BatchPipes subsystem.

cmdname
  is the name of the command that was entered.

parmname
  is the name of the keyword parameter that is too long.

Source: BATCHPIPES

Detecting Module:
ASFPMC05

System Action: Processing of the command is ended by the BatchPipes subsystem.

Operator Response: Re-enter the command again with a correct parameter value.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP202I BATCHPIPES *ssname cmdname* **COMMAND ERROR -- SPECIFIED** *parmname* **VALUE IS NOT VALID.**

Explanation: The *parmname* value that was specified on the command, *cmdname*, is not a valid parameter.

In the message text:

ssname
  is the name of the BatchPipes subsystem.

cmdname
  is the name of the command that was entered.

parmname
  is the name of the parameter that is in error.

Source: BATCHPIPES

Detecting Module:
ASFPMC05

System Action: Processing of the command is ended by the BatchPipes subsystem.

Operator Response: Re-enter the command again with a correct parameter value.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP203I BATCHPIPES *ssname cmdname* **COMMAND ERROR -- L= AREA ID IS NOT VALID.**

Explanation: The display area id indicated by the L= parameter on the command, *cmdname*, is not valid.

In the message text:

ssname
  is the name of the BatchPipes subsystem.

cmdname
  is the name of the command that was entered.

Source: BATCHPIPES

Detecting Module:
ASFPMC05

System Action: Processing of the command is ended by the BatchPipes subsystem.

Operator Response: Re-enter the command again with a valid L= parameter.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP204I BATCHPIPES *ssname cmdname* **COMMAND ERROR -- L= CONSOLE NAME IS NOT VALID.**

Explanation: The console name indicated by the L= parameter on the command, *cmdname*, is not valid.

In the message text:

ssname
is the name of the BatchPipes subsystem.

cmdname
is the name of the command that was entered.

Source: BATCHPIPES

Detecting Module:
ASFPMC05

System Action: Processing of the command is ended by the BatchPipes subsystem.

Operator Response: Re-enter the command again with a valid L= parameter.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP206I BATCHPIPES *ssname cmdname* **COMMAND ERROR -- NOT AUTHORIZED TO ISSUE COMMAND.**

Explanation: The operator/console is not authorized to enter the command, *cmdname*.

In the message text:

ssname
is the name of the BatchPipes subsystem.

cmdname
is the name of the command that was entered.

Source: BATCHPIPES

Detecting Module:
ASFPMC05

System Action: Processing of the command is ended by the BatchPipes subsystem.

Operator Response: Contact your installation's security administrator to ensure both you and the console are properly authorized to enter the command that you were attempting.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP207I BATCHPIPES *ssname cmdname* **COMMAND ERROR -- SPECIFIED KEYWORD IS NOT VALID.**

Explanation: The command entered, *cmdname*, specified a keyword that is not supported.

In the message text:

ssname
is the name of the BatchPipes subsystem.

cmdname
is the name of the command that was entered.

Source: BATCHPIPES

Detecting Module:
ASFPMC05

System Action: Processing of the command is ended by the BatchPipes subsystem.

Operator Response: If the keyword was misspelled, re-enter the command again with the correct keyword specified. Otherwise, if the keyword should not have been specified, re-enter the command without the unsupported keyword.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP209I SPECIFIED PIPE/JOB UNKNOWN TO *ssname*

Explanation: The pipe or job which was specified on the STATUS command is not known to the BatchPipes subsystem, *ssname*. In the case where status was requested for a pipe, the pipe specified is not currently allocated within the BatchPipes subsystem *ssname* for any job in the system and thus is not known to the subsystem. In the case where status was requested for a job, the specified job currently does not have a pipe allocated through the BatchPipes subsystem, *ssname*, and thus is not known to the subsystem.

In the message text:

ssname
is the name of the BatchPipes subsystem.

Source: BATCHPIPES

Detecting Module:
ASFPMCST

System Action: Processing of the STATUS command is completed.

Operator Response: If the jobname or pipe name/number was incorrectly entered on the STATUS command, re-enter the command with the correct specification.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP210I *hh.mm.ss ssname* **STATUS**

<Output for: **bp-cmd-prefix STATUS** without any parameters>

```
 #JOBS=nnnnn #PIPES=nnnnn #ALLOC=nnnnn STATUS=status MODE=mode
    #WAITALLOC=nnnnn       MAX WAITALLOC TIME=hh.mm.ss
#WRITERS=nnnnn
     #WAITOPEN=nnnnn        MAX WAITOPEN TIME=hh.mm.ss
         #IDLE=nnnnn            MAX IDLE TIME=hh.mm.ss
         #WAIT=nnnnn            MAX WAIT TIME=hh.mm.ss
    #WAITCLOSE=nnnnn       MAX WAITCLOSE TIME=hh.mm.ss
     #WAITTERM=nnnnn        MAX WAITTERM TIME=hh.mm.ss
#READERS=nnnnn
     #WAITOPEN=nnnnn        MAX WAITOPEN TIME=hh.mm.ss
         #IDLE=nnnnn            MAX IDLE TIME=hh.mm.ss
         #WAIT=nnnnn            MAX WAIT TIME=hh.mm.ss
      #WAITEOF=nnnnn         MAX WAITEOF TIME=hh.mm.ss
    #WAITCLOSE=nnnnn       MAX WAITCLOSE TIME=hh.mm.ss
     #WAITTERM=nnnnn        MAX WAITTERM TIME=hh.mm.ss
```

<Output for: **bp-cmd-prefix STATUS JOB=jobname|jobname*|*** >

```
JOB=jobname  S=stepname NUM=jobid
  PIPE=pipename                                DSPNM=xxxxxxxx
      WRITE   WAITOPEN=hh.mm.ss   COUNT=nnnnn   WAITS=nnnnn
JOB=jobname  S=stepname NUM=jobid
  PIPE=pipename                                DSPNM=xxxxxxxx
      READ        WAIT=hh.mm.ss   COUNT=nnnnn   WAITS=nnnnn
        ACCUM I/O WAIT=hh.mm.ss
  PIPE=pipename                                DSPNM=xxxxxxxx
      WRITE       IDLE=hh.mm.ss   COUNT=nnnnn   WAITS=nnnnn
        ACCUM I/O WAIT=hh.mm.ss
JOB=jobname  S=stepname NUM=jobid
  PIPE=pipename                                DSPNM=xxxxxxxx
      READ        IDLE=hh.mm.ss   COUNT=nnnnn   WAITS=nnnnn
        ACCUM I/O WAIT=hh.mm.ss
  PIPE=pipename                                DSPNM=xxxxxxxx
      READ    WAITCLOS=hh.mm.ss   COUNT=nnnnn   WAITS=nnnnn
        ACCUM I/O WAIT=hh.mm.ss
JOB=jobname  S=stepname NUM=jobid
  PIPE=pipename                                DSPNM=xxxxxxxx
      WRITE   WAITOPEN=hh.mm.ss   COUNT=nnnnn   WAITS=nnnnn
       REMAINING OPENS=(W=nnn,R=nnn)
JOB=jobname  S=stepname NUM=jobid
  PIPE=pipename                                DSPNM=xxxxxxxx
      READ     WAITEOF=hh.mm.ss   COUNT=nnnnn   WAITS=nnnnn
        ACCUM I/O WAIT=hh.mm.ss
         -
         -
```

<Output for: **bp-cmd-prefix STATUS JOB=jobname|jobname*|*,FLOW** >

```
PIPE=pipename                                  DSPNM=xxxxxxxx
  JOB=jobname  S=stepname NUM=jobid
      WRITE       WAIT=hh.mm.ss   COUNT=nnnnn   WAITS=nnnnn
        ACCUM I/O WAIT=hh.mm.ss
  JOB=jobname  S=stepname NUM=jobid
      READ    WAITOPEN=hh.mm.ss   COUNT=nnnnn   WAITS=nnnnn
  JOB=jobname  S=stepname NUM=jobid
      READ        IDLE=hh.mm.ss   COUNT=nnnnn   WAITS=nnnnn
        ACCUM I/O WAIT=hh.mm.ss
  JOB=jobname  S=stepname NUM=jobid
      READ        IDLE=hh.mm.ss   COUNT=nnnnn   WAITS=nnnnn
        ACCUM I/O WAIT=hh.mm.ss
         -
         -
```

<Output for: **bp-cmd-prefix STATUS PIPE=pipename|pipename*|*** >

```
PIPE=pipename                                  DSPNM=xxxxxxxx
  JOB=jobname  S=stepname NUM=jobid
      READ        IDLE=hh.mm.ss   COUNT=nnnnn   WAITS=nnnnn
        ACCUM I/O WAIT=hh.mm.ss
  JOB=jobname  S=stepname NUM=jobid
      WRITE       WAIT=hh.mm.ss   COUNT=nnnnn   WAITS=nnnnn
        ACCUM I/O WAIT=hh.mm.ss
  JOB=jobname  S=stepname NUM=jobid
      WRITE   WAITOPEN=hh.mm.ss   COUNT=nnnnn   WAITS=nnnnn
  JOB=jobname  S=stepname NUM=jobid
      WRITE   WAITCLOS=hh.mm.ss   COUNT=nnnnn   WAITS=nnnnn
        ACCUM I/O WAIT=hh.mm.ss
  JOB=jobname  S=stepname NUM=jobid
      WRITE    WAITEOF=hh.mm.ss   COUNT=nnnnn   WAITS=nnnnn
        ACCUM I/O WAIT=hh.mm.ss
  JOB=jobname  S=stepname NUM=jobid
      WRITE   WAITOPEN=hh.mm.ss   COUNT=nnnnn   WAITS=nnnnn
       REMAINING OPENS=(W=nnn,R=nnn)
         -
         -
```

<Output for: **bp-cmd-prefix STATUS TASK** >

```
TASK NAME=tttttttttttttttt  STATUS=ssssssssssssssss
         -
         -
```

n

<Output for: **bp-cmd-prefix STATUS PIPEPLEX** >

```
 PIPEPLEX-NAME: pipeplexname
 STRUCTURE-NAME: structurename
 PIPEPLEX-MEMBERS: nnn
    MEMBER=sysname  STATUS=plexstatus
 MAX-#PIPES-ALLOWED: nnn
 CUR-#PIPES-INUSE: nnn
 MAX-#PIPES-USED: nnn
 TOTAL-#PIPE-STORAGE-BLKS: nnn
 PIPE-STORAGE-BLKS-INUSE: nnn
```

n

Explanation: This message is issued in response to a BatchPipes STATUS command to display the status of one or more pipes or one or more jobs using pipes managed by the BatchPipes subsystem *ssname*.

Meanings of the various fields in the message text are as follows:

hh.mm.ss is the time the command was issued in hours (00-23), minutes (00-59), and seconds (00-59).

ssname is the name of the BatchPipes subsystem.

#JOBS=nnnnn indicates the number of unique jobs that are using (i.e., have allocated) at least one pipe data set.

#PIPES=nnnnn indicates the number of open pipe data sets. (NOTE: even though a pipe data set may be open under multiple jobs, it is only counted once in this number.)

#ALLOC=nnnnn
indicates the number of pipe data set allocations that have been done.

STATUS=status
Indicates the current status of the BatchPipes subsystem. The **status** may be one of the following:

ACTIVE indicates the subsystem is actively performing the BatchPipes subsystem functions.

STOPPING indicates the subsystem is ending its processing.

#WAITALLOC=nnnnn
indicates the number of jobs waiting in allocation for a corresponding partners to allocate the same pipe data set.

MAX WAITALLOC TIME=hh.mm.ss
the maximum time in hours (hh), minutes (mm), and seconds (ss) that any one job has been waiting in allocation for a corresponding partner to allocate the same pipe data set.

#WRITERS=nnnnn
indicates the current number of writers to one or more pipes. Each instance where a job is currently connected to a pipe for output is considered to be a writer.

#READERS=nnnnn
indicates the current number of readers of one or more pipes. Each instance where a job is currently connected to a pipe for input is considered to be a reader.

#WAITOPEN=nnnnn
Under **#WRITERS=nnnnn**, indicates the number of writers waiting in open for a corresponding reader partner to open the same pipe data set.

Under **#READERS=nnnnn**, indicates the number of readers waiting in open for a corresponding writer partner to open the same pipe data set.

MAX WAITOPEN TIME=hh.mm.ss
Under **#WRITERS=nnnnn**, is the maximum time in hours (hh), minutes (mm), and seconds (ss) that any one writer has been waiting in open for a corresponding reader partner to open the same pipe data set.

Under **#READERS=nnnnn**, is the maximum time in hours (hh), minutes (mm), and seconds (ss) that any one reader has been waiting in open for a corresponding writer partner to open the same pipe data set.

#IDLE=nnnnn Under **#WRITERS=nnnnn**, indicates the number of writers that are currently idle. A writer is said to be idle if it is not currently writing to the pipe (i.e., has no write request outstanding).

Under **#READERS=nnnnn**, indicates the number of readers that are currently idle. A reader is said to be idle if it is not currently reading from the pipe (i.e., has no read request outstanding).

MAX IDLE TIME=hh.mm.ss
Under **#WRITERS=nnnnn**, is the maximum time in hours (hh), minutes (mm), and seconds (ss) that a writer has ever been idle.

Under **#READERS=nnnnn**, is the maximum time in hours (hh), minutes (mm), and seconds (ss) that a reader has ever been idle.

#WAIT=nnnnn Under **#WRITERS=nnnnn**, indicates the number of writers waiting on a "pipe full" condition.

Under **#READERS=nnnnn**, indicates the number of readers waiting on a "pipe empty" condition.

MAX WAIT TIME=hh.mm.ss
Under **#WRITERS=nnnnn**, is the maximum time in hours (hh), minutes (mm), and seconds (ss) that any one writer has been waiting on a "pipe full" condition.

Under **#READERS=nnnnn**, is the maximum time in hours (hh), minutes (mm), and seconds (ss) that any one reader has been waiting on a "pipe empty" condition.

#WAITEOF=nnnnn
Under **#READERS=nnnnn**, indicates the number of readers waiting in a WAITEOF condition. All corresponding writer partners have

closed the same pipe data set, with the last writer closing the pipe specifying NOEOF.'

MAX WAITEOF TIME=hh.mm.ss
Under **#READERS=nnnnn**, is the maximum time in hours (hh), minutes (mm), and seconds (ss) that any one reader has been waiting in a WAITEOF condition.

#WAITCLOSE=nnnnn
Under **#WRITERS=nnnnn**, indicates the number of writers waiting in close for a corresponding partners to close the same pipe data set.

Under **#READERS=nnnnn**, indicates the number of readers waiting in close for a corresponding partners to close the same pipe data set.

MAX WAITCLOSE TIME=hh.mm.ss
Under **#WRITERS=nnnnn**, is the maximum time in hours (hh), minutes (mm), and seconds (ss) that any one writer has been waiting in close for corresponding partners to close the same pipe data set.

Under **#READERS=nnnnn**, is the maximum time in hours (hh), minutes (mm), and seconds (ss) that any one reader has been waiting in close for corresponding partners to close the same pipe data set.

#WAITTERM=nnnnn
Under **#WRITERS=nnnnn**, indicates the number of writers waiting in termination for a corresponding partners to terminate.

Under **#READERS=nnnnn**, indicates the number of readers waiting in termination for a corresponding partners to terminate.

MAX WAITTERM TIME=hh.mm.ss
Under **#WRITERS=nnnnn**, is the maximum time in hours (hh), minutes (mm), and seconds (ss) that any one writer has been waiting in termination for corresponding partners to terminate.

Under **#READERS=nnnnn**, is the maximum time in hours (hh), minutes (mm), and seconds (ss) that any one reader has been waiting in termination for corresponding partners to terminate.

JOB=jobname is the name of a job that is using a pipe data set (i.e., has at least allocated it).

S=stepname is the name of the current step within the job.

NUM=jobid is the job id (JOBnnnnn) that was assigned to the job.

PIPE=pipename
is the name of a pipe data set that has been opened.

DSPNM=xxxxxxxx
is the name of the data space that is being used for the pipe.

WRITE indicates the pipe data set is open for output.

READ indicates the pipe data set is open for input.

IDLE=hh.mm.ss
indicates the job is not currently reading or writing to the pipe data set even though it has the pipe data set open. The amount of time the job has been idle on the pipe data set is indicated in hours (hh), minutes (mm), and seconds (ss).

WAIT=hh.mm.ss
indicates the job is waiting on the pipe data set because of a "pipe full" condition (for a writer job) or a "pipe empty" condition (for a reader job). The amount of time the job has been waiting is indicated in hours (hh), minutes (mm), and seconds (ss).

WAITOPEN=hh.mm.ss
indicates the job is waiting in open for a corresponding partner job to open the same pipe data set. The amount of time the job has been waiting in open is indicated in hours (hh), minutes (mm), and seconds (ss).

WAITCLOS=hh.mm.ss
indicates the job is waiting in close for corresponding partner jobs to close the same pipe data set. The amount of time the job has been waiting in close is indicated in hours (hh), minutes (mm), and seconds (ss).

WAITEOF=hh.mm.ss
indicates the job is waiting in a WAITEOF condition, meaning that all writers have closed the pipe and the last writer closing specified the NOEOF subsystem parameter. The amount of time the job has been waiting in the WAITEOF condition is

indicated 'in hours (hh), minutes (mm), and seconds (ss).

COUNT=nnnnn is the number of blocks that have been read or written to the pipe data set thus far.

WAITS=nnnnn is the count of how many times the job has waited because of a "pipe full" condition (for a writer) or a "pipe empty" condition (for a reader).

ACCUM I/O WAIT=hh.mm.ss
indicates the total amount of time the job has spent waiting in a full or empty condition. The accumulated time is indicated in hours (hh), minutes (mm), and seconds (ss).

REMAINING OPENS=(w=n,r=n)
indicates that the job is waiting during open processing for more partners to connect to the pipe. The pipe has a specific number of writers and readers that must open the pipe before any connection is allowed to proceed. W=n and R=n describe the number of remaining connections of each type that must open the pipe before this connection is allowed to continue.'The wait-for-open time is indicated in hours (hh), minutes (mm), and seconds (ss).

FITTING indicates a fitting was specified on this connection.

Source: BATCHPIPES

Detecting Module:
ASFPMCST

System Action: BatchPipes subsystem processing continues.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP211I BATCHPIPES *ssname cmdname* COMMAND ERROR -- A PARAMETER VALUE CONTAINS A NON-NUMERIC CHARACTER

Explanation: The command entered, *cmdname*, specified a parameter value that was expected to be numeric, but a non-numeric character was found.

In the message text:

ssname
is the name of the BatchPipes subsystem.

cmdname
is the name of the command that was entered.

Source: BATCHPIPES

Detecting Module:
ASFPMC05

System Action: Processing of the command is ended by the BatchPipes subsystem.

Operator Response: Re-enter the command again with the correct parameter value specified.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP212I BATCHPIPES *ssname* TRACING {STARTED|STOPPED}

Explanation: Tracing by the subsystem *ssname* has been started or stopped as requested.

In the message text:

ssname
is the name of the BatchPipes subsystem.

STARTED
In response to a TRACE ALL, FLOW, or FUNCTION command, tracing has been started.

STOPPED
In response to a TRACE OFF command, tracing has been stopped.

Source: BATCHPIPES

Detecting Module:
ASFPMC05

System Action: BatchPipes subsystem processing continues.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP213I BATCHPIPES *ssname* TRACE STATUS: {INACTIVE|FLOW|FUNCTION|ALL}

Explanation: In response to a TRACE STATUS command, the status of trace activity for the subsystem *ssname* is indicated.

In the message text:

ssname
is the name of the BatchPipes subsystem.

INACTIVE
indicates tracing is not active.

FLOW
indicates FLOW tracing is being performed.

FUNCTION
indicates FUNCTION tracing is being performed.

ALL
indicates both FLOW and FUNCTION tracing is being performed.

Source: BATCHPIPES

Detecting Module:
ASFPMC05

System Action: BatchPipes subsystem processing continues.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP216I BATCHPIPES *ssname* STATUS COMMAND ERROR -- FLOW KEYWORD NOT ALLOWED WHEN WILD CARD CHARACTER USED.

Explanation: A STATUS command that was entered for the BatchPipes subsystem *ssname* specified both the FLOW keyword and a PIPE= or JOBNAME= parameter that contained a wild card character (e.g., *). This is not allowed.

In the message text:

ssname
is the name of the BatchPipes subsystem.

Source: BATCHPIPES

Detecting Module:
ASFPMC05

System Action: Processing of the command is ended by the BatchPipes subsystem.

Operator Response: Re-enter the command with either a specific jobname or pipename specified with the FLOW parameter, or re-enter the command without the FLOW parameter.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP217I BATCHPIPES *ssname* STOP INITIATED.

Explanation: In response to a STOP command, the BatchPipes subsystem *ssname* has initiated stop processing.

In the message text:

ssname
is the name of the BatchPipes subsystem.

Source: BATCHPIPES

Detecting Module:
ASFPMCST

System Action: The BatchPipes subsystem *ssname* will begin quiescing its processing in a nondisruptive manner. Any usage of pipes that is already in progress will be allowed to complete. When all pipe activity is completed, the BatchPipes subsystem will end normally.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP218I BATCHPIPES *ssname* CANCEL INITIATED.

Explanation: In response to a BatchPipes CANCEL command, the BatchPipes subsystem *ssname* has initiated cancel processing.

In the message text:

ssname
is the name of the BatchPipes subsystem.

Source: BATCHPIPES

Detecting Module:
ASFPMCST

System Action: The BatchPipes subsystem *ssname* will immediately cancel any jobs that currently have a pipe allocated which is managed by the subsystem. Also, no new allocations of pipes managed by the subsystem will be allowed. When all job cancellations have been initiated, the BatchPipes subsystem will then immediately end its processing.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP220I *hh.mm.ss ssname* **HELP INFO**

```
BATCHPIPES COMMAND SYNTAX:
 ... STatus
 ...
STatus,Job=jobname<*>|Pipe=pipename<*>
 ... STatus,J=job<,FLOW>
 ... STatus,P=pipe<,FLOW>
 ... STatus,TASK
 ... STOP
 ... SHUTDOWN
 ... CANCEL
 ... DUMP<,Job=jobname<*>>
<,Pipe=pipename<*>>
 ... DUMP<,Job=(jobname#1<*> <,...>
<,jobname#10<*>> )>
      <,Pipe=(pipename#1<*> <,...>
<,pipename#10<*>> )>
 ...
TRACE,ALL|FLOW|FUNCTION|OFF|STatus
 ... Display
 ... EOF,Pipe=pipename
 ... SET,MEMber=xx
 ... SET <,Idle=nnnn|OFF>
<,Wait=nnnn|OFF>
<,WaitOpen=nnnn|OFF>
      <,WaitClose=nnnn|OFF>
<,WaitEOF=nnnn|OFF>
      <,THResh=OFF|ON>
<,EOFRequired=YES|NO>
<,MAXBUFNO=nnn>
      <,DEFBUFNO=nnn>
```

Explanation: This message is a multiline message issued in response to the HELP command. It displays a summary of the syntax for the commands supported by the BatchPipes subsystem *ssname*.

In the message text:

hh.mm.ss
The time in hours (00-23), minutes (00-59), and seconds (00-59).

ssname
is the name of the BatchPipes subsystem.

Source: BATCHPIPES

Detecting Module:
ASFPMCST

System Action: BatchPipes subsystem processing continues.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP221I *hh.mm.ss* **DISPLAY**

<Output for: **bp-cmd-prefix DISPLAY** if threshold monitoring is active>

```
     ACTIVE PARMLIB MEMBER:  ASFPBPxx
     CURRENT THRESHOLD VALUES:
     IDLE=nnnn MINUTES   SOURCE:  setting-source
     WAIT=nnnn MINUTES   SOURCE:  setting-source
 WAITOPEN=nnnn MINUTES   SOURCE:  setting-source
WAITCLOSE=nnnn MINUTES   SOURCE:  setting-source
  WAITEOF=nnnn MINUTES   SOURCE:  setting-source
WAITALLOC=nnnn MINUTES   SOURCE:  setting-source
 WAITTERM=nnnn MINUTES   SOURCE:  setting-source
    CURRENT SUBSYSTEM SETTINGS:
    EOFREQUIRED=yes-no   SOURCE:  setting-source
    MAXBUFNO=nnn         SOURCE:  setting-source
    DEFBUFNO=nnn         SOURCE:  setting-source
    DB2SSID=ssid      DB2PLAN=planname
```

<Output for: **bp-cmd-prefix DISPLAY** if threshold monitoring is inactive>

```
BATCHPIPES/MVS THRESHOLD MONITORING IS NOT ACTIVE
CURRENT SUBSYSTEM SETTINGS:
EOFREQUIRED=yes-no   SOURCE:  setting-source
MAXBUFNO=nnn         SOURCE:  setting-source
DEFBUFNO=nnn         SOURCE:  setting-source
```

Explanation: This message is issued in response to a BatchPipes DISPLAY command. The current active BatchPipes PARMLIB member and the current subsystem settings are displayed.

Meanings of the various fields in the message text are as follows:

hh.mm.ss is the time the command was issued in hours (00-23), minutes (00-59), and seconds (00-59).

ssname is the name of the BatchPipes subsystem.

ACTIVE PARMLIB MEMBER: ASFPBPxx
identifies the most recent BatchPipes PARMLIB member, **ASFPBPxx**, that was used to set BatchPipes processing parameters.

IDLE=nnnn MINUTES
identifies the current IDLE inactivity threshold value, **nnnn**, in minutes. When a job has a pipe data set open but has not performed a read or a write operation to the pipe data set for a period that exceeds the IDLE inactivity threshold, the operator is notified.

WAIT=nnnn MINUTES
identifies the current WAIT inactivity threshold value, **nnnn**, in minutes. When a job is waited because of either a "pipe full" condition or a "pipe empty" condition and the period waiting exceeds the WAIT inactivity threshold, the operator is notified.

WAITOPEN=nnnn MINUTES
identifies the current WAITOPEN inactivity threshold value, **nnnn**, in minutes. When a reader or writer job opening a pipe data set is forced to wait for a corresponding partner job to also open the pipe data set and the time waiting exceeds the WAITOPEN inactivity threshold, the operator is notified.

WAITCLOSE=nnnn MINUTES
identifies the current WAITCLOSE inactivity threshold value, **nnnn**, in minutes. When a reader or writer job closing a pipe data set is forced to wait for corresponding partner jobs to also close the pipe data set and the time waiting exceeds the WAITCLOSE inactivity threshold, the operator is notified.

WAITEOF=nnnn MINUTES
identifies the current WAITEOF inactivity threshold value, **nnnn**, in minutes. When a reader job waiting on an empty pipe data set has no corresponding writer jobs with the pipe data set open (the writer jobs have all closed the pipe data set, and the last writer closed the pipe specifying NOEOF) and the time waiting exceeds the WAITEOF inactivity threshold, the operator is notified.

WAITALLOC=nnnn MINUTES
identifies the current WAITALLOC inactivity threshold value, **nnnn**, in minutes. When a reader or writer job allocating a pipe data set is forced to wait for corresponding partner jobs to also allocate the pipe data set and the time waiting exceeds the WAITALLOC inactivity threshold, the operator is notified.

WAITTERM=nnnn MINUTES
identifies the current WAITTERM inactivity threshold value, **nnnn**, in minutes. When a reader or writer job is terminating and is forced to wait for corresponding partner jobs to also terminate and the time waiting exceeds the WAITTERM inactivity threshold, the operator is notified.

EOFRequired=YESlNO
identifies the current EOFRequired subsystem setting. Reader connections closing a pipe data set before receiving EOF will trigger an error on the pipe data set if EOFRequired=YES is specified. Specifying EOFRequired=NO will allow reader connections to close the pipe data set without receiving EOF. This value is in effect when no overriding subsystem parameter is specified by the reader job.'

MAXBUFNO=nnn
identifies the current maximum pipe depth (BUFNO) allowed on a pipe.

DEFBUFNO=nnn
identifies the current default pipe depth (BUFNO) for a pipe.

DB2SSID=ssid
identifies the name of the DB2 subsystem to
used by BatchPipeWorks.

DB2PLAN=planname
identifies the DB2 Plan to be used by BatchPipeWorks.

SOURCE: setting-source
identifies the source that was used to establish the subsystem setting value. The **setting-source** may be one of the following:

COMMAND
indicates the subsystem setting value was set using the SET command.

PARMLIB
indicates the subsystem setting value was set from the BatchPipes PARMLIB member.

DEFAULT
indicates the subsystem setting value was set using the BatchPipes default value.

Source: BATCHPIPES

Detecting Module:
ASFPMCST

System Action: BatchPipes subsystem processing continues.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: If the display command indicates that BatchPipes threshold monitoring is inactive, the threshold detection task has abnormally ended. Obtain the SYSLOG and SVC Dump associated with the abend and contact the IBM Support Center.

ASFP230I BATCHPIPES ***ssname*** **SET COMMAND ERROR - OUT OF RANGE VALUE SPECIFIED FOR THE** ***pname*** **PARAMETER.**

Explanation: The BatchPipes SET command was issued with an invalid value specified for the indicated parameter. The value is out of the range of values supported.

In the message text:

ssname
is the name of the BatchPipes subsystem.

pname
One of the following:

IDLE
indicates the read/write IDLE inactivity threshold is in error. Valid values are from 0 to 1440, or OFF.

WAIT
indicates the read/write WAIT inactivity threshold is in error. Valid values are from 0 to 1440, or OFF.

WAITOPEN
indicates the read/write WAITOPEN inactivity threshold is in error. Valid values are from 0 to 1440, or OFF.

WAITCLOSE
indicates the job WAITCLOSE inactivity threshold is in error. Valid values are from 0 to 1440, or OFF.

WAITEOF
indicates the read WAITEOF inactivity threshold is in error. Valid values are from 0 to 1440, or OFF.

WAITALLOC
indicates the job WAITALLOC inactivity threshold is in error. Valid values are from 0 to 1440,
or OFF.

WAITTERM
indicates the job WAITTERM inactivity threshold is in error. Valid values are from 0 to 1440,
or OFF.

MAXBUFNO
indicates the job MAXBUFNO value specified is out of range. Valid values are from 1 to 255.

DEFBUFNO
indicates the job DEFBUFNO value specified is out of range. Valid values are from 1 to 255.

DB2SSID
indicates the value specified for the subsystem parameter DB2SSID is too long. The subsystem id may be from 1 to 4 characters.

DB2PLAN
indicates the value specified for the subsystem parameter DB2PLAN is too long. The plan name may be from 1 to 8 characters.

Source: BATCHPIPES

Detecting Module:
ASFPMC05

System Action: The indicated parameter remains unchanged.

Operator Response: Re-enter the BatchPipes SET command with a correct value specified for the parameter.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP231I BATCHPIPES ***ssname*** **SET COMMAND ERROR - NAME TOKEN SERVICES ERROR PROCESSING THE** ***pname*** **PARAMETER.**

Explanation: The BatchPipes SET command was issued with a parameter that attempted to use name/token services. Either the load failed for the name/token services or the service issued a non-zero return code.

In the message text:

ssname
is the name of the BatchPipes subsystem.

pname
One of the following:

DB2SSID
indicates the DB2SSID parameter was specified.

DB2PLAN
indicates the DB2PLAN parameter was specified.

Source: BATCHPIPES

Detecting Module:
ASFPMC05

System Action: The BatchPipe default setting remains unchanged.

Operator Response: DB2SSID and DB2PLAN keywords are only supported on releases of MVS that support name/token services.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP240I BATCHPIPES *ssname* EOF COMMAND ERROR. PIPE *pname* IS NOT DORMANT.

Explanation: Pipe *pname* was not dormant when the EOF command attempted to present EOF on the pipe. A pipe is dormant when all writers have closed the pipe, and the last writer that closed the pipe specified the *NOEOF* subsystem parameter.

In the message text:

ssname
is the name of the BatchPipes subsystem.

pname
is the name of the pipe data set.

Source: BATCHPIPES

Detecting Module:
ASFPMCEO

System Action: The EOF command ends without presenting EOF to the readers on the pipe.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: Issue the BatchPipes STATUS PIPE=*pname* command to see all of the jobs currently using the pipe. It is possible that the pipe will close normally, since it is not yet dormant. Wait until the pipe becomes dormant to issue the EOF command again, or allow the pipe to end normally.

ASFP241I BATCHPIPES *ssname* EOF SUCCESSFUL. EOF PRESENTED ON PIPE *pname*

Explanation: Pipe *pname* was dormant when the EOF command was issued, and EOF was presented to the pipe's reader connections. A pipe is dormant when all writers have closed the pipe, and the last writer that closed the pipe specified the *NOEOF* subsystem parameter.

In the message text:

ssname
is the name of the BatchPipes subsystem.

pname
is the name of the pipe data set.

Source: BATCHPIPES

Detecting Module:
ASFPMCEO

System Action: The EOF command ends successfully.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP242I BATCHPIPES *ssname* EOF COMMAND ERROR. PIPE *pname* NOT FOUND

Explanation: Pipe *pname* was not found when the EOF command searched the BatchPipes *ssname* subsystem.

In the message text:

ssname
is the name of the BatchPipes subsystem.

pname
is the name of the pipe data set.

Source: BATCHPIPES

Detecting Module:
ASFPMCEO

System Action: The EOF command ends.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: Check the command and determine if the *pname* was incorrectly specified. Ensure that the EOF command was directed to the correct BatchPipes subsystem. If no errors are detected in the command you issued, try the BatchPipes STATUS P=*pname* to determine if the jobs processing the pipe ended before the EOF command was issued.

ASFP243I BATCHPIPES *ssname* EOF COMMAND FAILURE. A CATASTROPHIC SUBSYSTEM ERROR OCCURRED.

Explanation: The BatchPipes/MVS EOF command was issued and during the processing of that command, a subsystem error was encountered. A catastrophic error condition occurred under the BatchPipes subsystem address space that prevents it from processing any further requests. The subsystem has either lost a critical service required for processing or has encountered a critical error with the BatchPipe coupling facility structure.

In the message text:

ssname
is the name of the BatchPipes subsystem.

Source: BATCHPIPES

Detecting Module:
ASFPMXEO

System Action: The command processing is terminated.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and will need to be rescheduled, have the operator cancel these jobs. Notify your system programmer.

Once the problem is corrected, do any data set cleanup that may be required to rerun any jobs affected by the failure.

System Programmer Response: Examine the SYSLOG at the time of the job failure. If the BatchPipes subsystem on the system where the job was running lost a critical service task due to an abnormal termination, obtain the SVC dump associated with abend and determine the cause of the problem. On the other hand, if the SYSLOG at the time of the ,'job failure shows that message ASFP429E was issued indicating a BatchPipes coupling facility structure failure or connectivity problem occurred, determine the cause of the problem. If the problem is due to an installation error, correct the problem and then notify the application programmer responsible for the the pipeline so that all affected jobs may be rescheduled. Otherwise, contact the IBM Support Center.

ASFP310I BATCHPIPES ALLOCATION FAILURE: ABEND=*abendcode* **RSN=***rsncode* **JOB=***jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname*

Explanation: An unexpected abnormal end occurred during BatchPipes allocation processing for the pipe data set defined by the DD statement, *ddname*.

In the message text:

abendcode
is the system abend completion code.

rsncode
is the abend reason code, or zero if there is no reason code associated with the system abend.

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being allocated.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSAU

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and need to be rescheduled, have the operator cancel these jobs. Then determine the cause of the failure. If the failure is because of an application error correct the problem. Otherwise, notify your system programmer.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: Obtain the SYSLOG and the SVC Dump associated with the indicated abend and determine the cause of the problem. If the failure is due to an installation error, correct the problem and then notify the application programmer responsible for the pipeline so that all affected jobs can be rescheduled. Otherwise, contact the IBM Support Center.

ASFP311I PROCESSING ENDED FOR JOB IN ALLOCSYNC WAIT: JOB=*jobstep* **STEP=***jobstep* **SUBSYS=***ssname*

Explanation: A job in BatchPipes AllocSync wait was ended during allocation processing for the BatchPipes subsystem indicated.

In the message text:

jobstep
is the name of the job that failed.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSAA

System Action: None.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP312I BATCHPIPES UNALLOCATION FAILURE: ABEND=*abendcode* **RSN=***rsncode* **JOB=***jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname*

Explanation: An unexpected abnormal end occurred during BatchPipes unallocation processing for the pipe data set defined by the DD statement, *ddname*.

In the message text:

abendcode
is the abend completion code.

rsncode
is the abend reason code, or zero if there is no reason code associated with the abend.

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being unallocated.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSAU

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and need to be rescheduled, have the operator cancel these jobs. Then determine the cause of the failure. If the failure is because of an application error, correct the problem. Otherwise, notify your system programmer.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: Obtain the SYSLOG and the SVC Dump associated with the indicated abend and determine the cause of the problem. If the failure is due to an installation error, correct the problem and then notify the application programmer responsible for the pipeline so that all affected jobs can be rescheduled. Otherwise, contact the IBM Support Center.

ASFP314I BATCHPIPES ALLOCATION FAILURE. SUBSYSTEM NOT ACTIVE. JOB=*jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname*

Explanation: Allocation processing for the pipe data set defined by the DD statement, *ddname*, failed because the BatchPipes subsystem specified by the DD statement is no longer active.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being allocated.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSAA

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure and do any data set cleanup that may be required to rerun the jobs. Then contact your operator or system programmer to ensure the BatchPipes subsystem has been restarted.

When the BatchPipes subsystem has been restarted, rerun the job that failed and all other jobs in the pipeline that were affected by the failure.

System Programmer Response: None.

ASFP315I BATCHPIPES UNALLOCATION FAILURE. SUBSYSTEM NOT ACTIVE: JOB=*jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname*

Explanation: Unallocation processing for the pipe data set defined by the DD statement, *ddname*, failed because the BatchPipes subsystem specified by the DD statement is no longer active.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being unallocated.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSAU

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure and do any data set cleanup that may be required to rerun the jobs. Then contact your operator or system programmer to ensure the BatchPipes subsystem has been restarted.

When the BatchPipes subsystem has been restarted, rerun the job that failed and all other jobs in the pipeline that were affected by the failure.

System Programmer Response: None.

ASFP318I BATCHPIPES ALLOCATION FAILURE. SUBSYSTEM IS ENDING. JOB=*jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname*

Explanation: Allocation processing for the pipe data set defined by the DD statement, *ddname*, failed because the BatchPipes subsystem specified on the DD statement, is in the process of ending.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being allocated.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSAA

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure and do any data set cleanup that may be required to rerun the jobs. Then contact your operator or system programmer to ensure the BatchPipes subsystem has been restarted.

When the BatchPipes subsystem has been restarted, rerun the job that failed and all other jobs in the pipeline that were affected by the failure.

System Programmer Response: None.

ASFP320I BATCHPIPES FAILURE. RECORD FORMAT IS NOT VALID. JOB=*jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname*

Explanation: Processing for the pipe data set defined by the DD statement, *ddname*, failed because the record format specified is not supported by BatchPipes. The RECFM parameter must be either F, FA, FM, FB, FBA, FBM, FS, FSA, FSM, FBS, FBSA, FBSM, V, VA, VM, VB, VBA or VBM.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being processed.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSAA ASFPMSOO

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and need to be rescheduled, have the operator cancel these jobs. Then correct the RECFM parameter specified on the DD statement for the pipe data set.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: None.

ASFP321I BATCHPIPES FAILURE. RECFM NOT SPECIFIED. JOB=*jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname*

Explanation: Processing for the pipe data set defined by the DD statement, *ddname*, failed because no RECFM parameter was specified on the DD statement.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being processed.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSAA ASFPMSOO

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and need to be rescheduled, have the operator cancel these jobs. Then correct the DD statement for the pipe data set so it specifies a valid RECFM parameter value (i.e., F, FA, FM, FB, FBA, FBM, FS, FSA, FSM, FBS, FBSA, FBSM, V, VA, VM, VB, VBA, VBM).

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: None.

ASFP322I BATCHPIPES FAILURE. LRECL NOT SPECIFIED. JOB=*jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname*

Explanation: Processing for the pipe data set defined by the DD statement, *ddname*, failed because no LRECL parameter was specified on the DD statement.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being processed.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSAA ASFPMSOO

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and need to be rescheduled, have the operator cancel these jobs. Then correct the DD statement for the pipe data set so it specifies a valid LRECL parameter value.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: None.

ASFP324I BATCHPIPES FAILURE. LRECL GREATER THAN BLKSIZE. JOB=*jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname*

Explanation: Processing for the pipe data set defined by the DD statement, *ddname*, failed because the LRECL parameter specified on the DD statement is larger than the BLKSIZE parameter that was specified.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being processed.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSAA ASFPMSOO

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and need to be rescheduled, have the operator cancel these jobs. Then correct the DD statement for the pipe data set so it specifies a valid LRECL size.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: None.

ASFP325I BATCHPIPES FAILURE. LRECL=X SPECIFIED. JOB=*jobname* STEP=*jobstep* DD=*ddname* SUBSYS=*ssname*

Explanation: Processing for the pipe data set defined by the DD statement, *ddname*, failed because the character "X" was incorrectly specified as the LRECL value.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being processed.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSAA ASFPMSOO

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and need to be rescheduled, have the operator cancel these jobs. Then correct the DD statement for the pipe data set so it specifies a valid LRECL size.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: None.

ASFP326I BATCHPIPES SUBSYSTEM *ssname* ALTERED TO USE THE PRIMARY SUBSYSTEM.

Explanation: The BatchPipes subsystem, *ssname*, has been set to indicate that the it is to be started only under the primary subsystem.

This message occurs if the installation has specified the BatchPipes subsystem initialization routine, ASFPMSSI, on the initialization statement for the BatchPipes subsystem in its IEFSSNxx PARMLIB member. The BatchPipes subsystem initialization routine will always force the specified BATCHPIPES subsystem is to be initialized under only the primary subsystem.

This message is written to hard copy only.

In the message text:

ssname
is the name of the BatchPipes subsystem.

Source: BATCHPIPES

Detecting Module:
ASFPMSSI

System Action: System initialization continues.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP327I BATCHPIPES FAILURE. BLKSIZE IS NOT CONSISTENT WITH PIPE. JOB=*jobname* STEP=*jobstep* DD=*ddname* SUBSYS=*ssname*

Explanation: Processing for the pipe data set defined by the DD statement, *ddname*, failed because the BLKSIZE parameter specified on the DD statement does not match the BLKSIZE of the existing pipe partner.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being processed.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSOO

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine whether this job should connect to the pipe data set identified by the DSN= and SUBSYS= keywords. If not, correct the DSN= or SUBSYS= keyword(s) to identify the correct pipe data set and BatchPipes subsystem. Otherwise determine whether this DD statement or the partner job DD statement is using the correct BLKSIZE, and make all partners conform to correct and consistent BLKSIZE.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: None.

ASFP328I BATCHPIPES FAILURE. LRECL IS NOT CONSISTENT WITH PIPE. JOB=*jobname* STEP=*jobstep* DD=*ddname* SUBSYS=*ssname*

Explanation: Processing for the pipe data set defined by the DD statement, *ddname*, failed because the LRECL parameter specified on the DD statement does not match the LRECL of the existing pipe partner.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being processed.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSOO

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine whether this job should connect to the pipe data set identified by the DSN= and SUBSYS= keywords. If not, correct the DSN= or SUBSYS= keyword(s) to identify the correct pipe data set and BatchPipes subsystem. Otherwise determine whether this DD statement or the partner job DD statement is using the correct LRECL, and make all partners conform to correct and consistent LRECL.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: None.

ASFP329I BATCHPIPES FAILURE. RECFM IS NOT CONSISTENT WITH PIPE. JOB=*jobname* STEP=*jobstep* DD=*ddname* SUBSYS=*ssname*

Explanation: Processing for the pipe data set defined by the DD statement, *ddname*, failed because the RECFM parameter specified on the DD statement does not match the RECFM of the existing pipe partner.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being processed.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSOO

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine whether this job should connect to the pipe data set identified by the DSN= and SUBSYS= keywords. If not, correct the DSN= or SUBSYS= keyword(s) to identify the correct pipe data set and BatchPipes subsystem. Otherwise determine whether this DD statement or the partner job DD statement is using the correct RECFM, and make all partners conform to correct and consistent RECFM.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: None.

ASFP330I BATCHPIPES CONVERTER PROCESSING FAILURE: ABEND=*abendcode* RSN=*rsncode* DD=*ddname* SUBSYS=*ssname*

Explanation: An unexpected abnormal end occurred during BatchPipes converter processing for the DD statement, *ddname*.

In the message text:

abendcode
is the abend completion code.

rsncode
is the abend reason code, or zero if there is no reason code associated with the abend.

ddname
is the name of the DD statement being converted.

ssname
is the name of the BatchPipes subsystem specified by the DD.

Source: BATCHPIPES

Detecting Module:
ASFPMSC

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and need to be rescheduled, have the operator cancel these jobs. Then determine the cause of the failure. If the failure is because of an application error, correct the problem. Otherwise, notify your system programmer.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: Obtain the SYSLOG and the SVC Dump associated with the indicated abend and determine the cause of the problem. If the failure is due to an installation error, correct the problem and then notify the application programmer responsible for the pipeline so that all affected jobs may be rescheduled. Otherwise, contact the IBM Support Center.

ASFP331I INSUFFICIENT AUTHORITY: *jobname*, *jobstep*, *ddname*, *ssname*, *rname*

Explanation: Open processing for the pipe defined by the DD statement, *ddname*, failed because of insufficient access authority to the resource, *rname*. A fitting was specified on the DD statement. However, the user is not properly authorized to access fittings for BatchPipes.

In the message text:

jobname
is the name of the failing job.

jobstep
is the name of the failing job step.

ddname
is the DD name of the pipe data set that was being opened

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

rname
is the resource name that failed the authority check.

Source: BATCHPIPES

Detecting Module:
ASFPMSSC

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system, have the operator cancel these jobs. Check to ensure that the correct subsystem name was specified on the DD statement that failed authorization. If the subsystem name was incorrect, specify the correct name. Otherwise, contact your system security administrator to define and obtain proper authorization to the resource.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: None.

ASFP332I REQUIRED RESOURCE CHECK FAILED: *jobname*, *jobstep*, *ddname*, *ssname*, *rname*

Explanation: Open processing for the pipe defined by the DD statement, *ddname*, failed because the System Authorization Facility (SAF) did not verify the user's authority to use *rname*. The user specified a fitting that was to be used with an authorized program, which must successfully pass a SAF authority check. 'Either SAF made no decision on the authority check or the 'resource was not defined to the security product.

In the message text:

jobname
is the name of the failing job.

jobstep
is the name of the failing job step.

ddname
is the DD name of the pipe data set that was being opened

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

rname
is the resource name that was to be used for the authority check.

Source: BATCHPIPES

Detecting Module:
ASFPMSSC

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system, have the operator cancel these jobs. Check to ensure that the correct subsystem name was specified on the DD statement that failed authorization. If the subsystem name was incorrect, specify the correct name. Otherwise, contact your system security administrator to obtain proper authorization to the resource.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: None.

ASFP333I INSUFFICIENT AUTHORITY: *jobname***,** *jobstep***,** *ddname***,** *ssname***,** *rname*

Explanation: Open processing for the pipe defined by the DD statement, *ddname*, failed because of insufficient access authority to the resource, *rname*. A fitting was specified on the DD statement, which was opened by an authorized program. To use fittings on authorized programs, the user must have access to resource *rname*.

In the message text:

jobname
is the name of the failing job.

jobstep
is the name of the failing job step.

ddname
is the DD name of the pipe data set that was being opened

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

rname
is the resource name that failed the authority check.

Source: BATCHPIPES

Detecting Module:
ASFPMSSC

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system, have the operator cancel these jobs. Check to ensure that the correct subsystem name was specified on the DD statement that failed authorization. If the subsystem name was incorrect, specify the correct name. Otherwise, contact your system security administrator to obtain proper authorization to the resource.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: None.

ASFP334I BATCHPIPES FAILURE. SUBSYSTEM NOT ABLE TO ALLOCATE PIPE. JOB=*jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname*

Explanation: Processing for the pipe data set defined by the DD statement, *ddname*, failed because the BatchPipes coupling facility structure which is used to manage the pipe as a cross-system pipe, is full. As a result, the job, *jobname*, cannot be connected to the pipe.

The name of BatchPipes coupling facility structure that is full is SYSASFPssnm where ssnm is the BatchPipes subsystem name.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being allocated.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSAA

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and will need to be rescheduled, have the operator cancel these jobs. Then notify your system programmer that there was not enough space in the BatchPipes coupling facility to allow your job to allocate the pipe and identify the name of the structure.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: Increase the size of the BatchPipes coupling facility structure in your installation's Coupling Facility Resource Management (CFRM) policy and restart the entire BatchPipes pipeplex which was using the structure. When the pipeplex is restarted, notify the application programmer responsible for the pipeline so that all affected jobs may be rescheduled.

ASFP335I BATCHPIPES OPEN FAILURE. THE JOB IS ALREADY CONNECTED TO THE PIPE. JOB=*jobname* STEP=*jobstep* DD=*ddname* SUBSYS=*ssname*

Explanation: OPEN processing for the pipe data set defined by the DD statement, *ddname*, failed because the job is already connected to the specified pipe for the same type of processing (i.e., read or write).

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being opened.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSOO

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and need to be rescheduled, have the operator cancel these jobs. Then determine why the job that failed is opening the indicated pipe more than once for read or write and correct the problem. When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: None.

ASFP336I BATCHPIPES FAILURE. MAXIMUM NUMBER OF PIPES IN USE. JOB=*jobname* STEP=*jobstep* DD=*ddname* SUBSYS=*ssname*

Explanation: Processing for the pipe data set defined by the DD statement, *ddname*, failed because the BatchPipes Pipeplex the pipe is associated with is already at the maximum number of cross-system pipes allowed by the defined BatchPipes coupling facility structure.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being allocated.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSAA

System Action: The job is abnormally ended.

Operator Response: Notify your system programmer.

User Response: None.

Application Programmer Response: Notify your system programmer that there were not enough pipes available in the BatchPipes pipeplex to run your pipeline of jobs. Provide your system programmer with the name of the pipeplex. The name is the same as the BatchPipes subsystem name that was specified on the DD statement for the pipe that failed.

Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and will need to be rescheduled, have the operator cancel these jobs. Then either wait for the pipe usage in the BatchPipes pipeplex to decline so that there are enough pipes available for your pipeline or wait for your system programmer to adjust the pipeplex so it supports a larger number of pipes. When there are sufficient pipes available, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: Re-estimate the maximum number of pipes that need to be used concurrently in the pipeplex where the failed job was running and recalculate the size of the BatchPipes coupling facility structure that is required to support the new maximum number of pipes. Then update the size of the structure in the Coupling Facility Resource Management (CFRM) policy. The name of the structure associated with the pipeplex should be SYSASFPssnm where ssnm is the subsystem name. When you have updated the size of the structure, stop the current pipeplex and then restart the pipeplex with the new structure size.

ASFP337I BATCHPIPES FAILURE. MAXIMUM CONNECTIONS TO PIPE EXCEEDED. JOB=*jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname*

Explanation: Processing for the pipe data set defined by the DD statement, *ddname*, failed because the specified pipe is already at the maximum number of job connections allowed.

This message is issued only in the case where the pipe is operating in cross-system mode and has reached its maximum number of job connections. For a cross-system pipe, the maximum number of job connections allowed is 32.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being allocated.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSAA

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and will need to be rescheduled, have the operator cancel these jobs. Then correct the problem by reducing the number of jobs in your BatchPipes pipeline that concurrently use the pipe the failed open processing. Ensure that no more than 32 jobs use the pipe concurrently. When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: None.

ASFP338I BATCHPIPES FAILURE. ABEND=*abendcode* **RSN=***rsncode* **JOB=***jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname*

Explanation: Processing for the pipe data set defined by the DD statement, *ddname*, failed because an abnormal end occurred when the BatchPipes subsystem address space was processing the request.

In the message text:

abendcode
is the abend completion code.

rsncode
is the abend reason code, or zero if there is no reason code associated with the abend.

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being processed.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSAA ASFPMSOO

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and will need to be rescheduled, have the operator cancel these jobs. Then notify your system programmer.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: Obtain the SYSLOG and the SVC Dump associated with the indicated abend and determine the cause of the problem. If the failure is due to an installation error, correct the problem and then notify the application programmer responsible for the pipeline so that all affected jobs can be rescheduled. Otherwise, contact the IBM Support Center.

ASFP339I BATCHPIPES OPEN FAILURE.
A CATASTROPHIC SUBSYSTEM ERROR OCCURRED.
THE SUBSYSTEM IS NO LONGER ABLE TO OPEN PIPES.
JOB=*jobname* **STEP=***jobstep*
DD=*ddname* **SUBSYS=***ssname*

Explanation: OPEN processing for the pipe data set defined by the DD statement, *ddname*, failed because the BatchPipes subsystem *ssnm* is no longer able to perform open processing. A catastrophic error condition occurred under the BatchPipes subsystem address space that prevents it from processing any further OPEN requests. The subsystem has either lost a critical service required for OPEN processing or has encountered a critical error with the BatchPipes coupling facility structure.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being opened.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSOO

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and will need to be rescheduled, have the operator cancel these jobs. Then notify your system programmer.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: Examine the SYSLOG at the time of the job failure. If the BatchPipes subsystem on the system where the job was running lost a critical service task because of an abnormal end, obtain the SVC dump associated with abend and determine the cause of the problem. On the other hand, if the SYSLOG at the time of the ,'job failure shows that message ASFP429E was issued indicating a BatchPipes coupling facility structure failure or connectivity problem occurred, determine the cause of the problem. If the problem is due to an installation error, correct the problem and then notify the application programmer responsible for the pipeline so that all affected jobs can be rescheduled. Otherwise, contact the IBM Support Center.

ASFP340I BATCHPIPES CHECKPOINT FAILURE:
ABEND=*abendcode* **RSN=***rsncode*
JOB=*ssname* **STEP=***ssname*
DD=*ssname* **SUBSYS=***ssname*

Explanation: An unexpected abnormal end occurred during BatchPipes checkpoint processing for the job in which this message appears.

In the message text:

abendcode
is the abend completion code.

rsncode
is the abend reason code, or zero if there is no reason code associated with the abend.

ssname
is the name of the job subsystem that was performing the checkpoint processing when the failure occurred.

Source: BATCHPIPES

Detecting Module:
ASFPMSCK

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and need to be rescheduled, have the operator cancel these jobs. Then determine the cause of the failure. If the failure was caused by an application error correct the problem. Otherwise, notify your system programmer.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: Obtain the SYSLOG and the SVC Dump associated with the indicated abend and determine the cause of the problem. If the failure is due to an installation error, correct the problem and then notify the application programmer responsible for the pipeline so that all affected jobs can be rescheduled. Otherwise, contact the IBM Support Center.

ASFP341I BATCHPIPES CHECKPOINT RESTART FAILURE: ABEND=*abendcode* RSN=*rsncode* JOB=*ssname* STEP=*ssname* DD=*ssname* SUBSYS=*ssname*

Explanation: An unexpected abnormal end occurred during BatchPipes checkpoint restart processing for the job in which this message appears.

In the message text:

abendcode
is the abend completion code.

rsncode
is the abend reason code, or zero if there is no reason code associated with the abend.

ssname
is the name of the job subsystem that was performing the restart processing when the failure occurred.

Source: BATCHPIPES

Detecting Module:
ASFPMSRS

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and need to be rescheduled, have the operator cancel these jobs. Then determine the cause of the failure. If the failure is due to an application error correct the problem. Otherwise, notify your system programmer.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: Obtain the SYSLOG and the SVC Dump associated with the indicated abend and determine the cause of the problem. If the failure is due to an installation error, correct the problem and then notify the application programmer responsible for the pipeline so that all affected jobs can be rescheduled. Otherwise, contact the IBM Support Center.

ASFP342I BATCHPIPES CLOSE FAILURE. A CATASTROPHIC SUBSYSTEM ERROR OCCURRED. THE SUBSYSTEM IS NO LONGER ABLE TO CLOSE PIPES. JOB=*jobname* STEP=*jobstep* DD=*ddname* SUBSYS=*ssname*

Explanation: CLOSE processing for the pipe data set defined by the DD statement, *ddname*, failed because the BatchPipes subsystem *ssnm* is no longer able to perform close processing. A catastrophic error condition occurred under the BatchPipes subsystem address space that prevents it from processing any further CLOSE requests. The subsystem has either lost a critical service required for CLOSE processing or has encountered a critical error with the BatchPipes coupling facility structure.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being opened.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSOC

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and will need to be rescheduled, have the operator cancel these jobs. Then notify your system programmer.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: Examine the SYSLOG at the time of the job failure. If the BatchPipes subsystem on the system where the job was running lost a critical service task due to an abnormal end, obtain the SVC dump associated with abend and determine the cause of the problem. On the other hand, if the SYSLOG at the time of the ,'job failure shows that message ASFP429E was issued indicating a BatchPipes coupling facility structure failure or connectivity problem occurred, determine the cause of the problem. If the problem is due to an installation error, correct the problem and then notify the application programmer responsible for the pipeline so that all affected jobs can be rescheduled. Otherwise, contact the IBM Support Center.

ASFP343I BATCHPIPES CLOSE FAILURE. UNABLE TO CLOSE PIPE DUE TO A PREVIOUS CLOSE FAILURE. JOB=*jobname* STEP=*jobstep* DD=*ddname* SUBSYS=*ssname*

Explanation: The BatchPipes subsystem *ssnm* when processing a CLOSE request for the pipe data set defined by the DD statement, *ddname*, determined that there was a prior attempt to CLOSE the pipe data set that failed and left the pipe control structure in an unpredictable state. As a result, subsequent CLOSE requests cannot be performed.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being opened.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSOC

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Examine the job log for the error message describing the previous failure that occurred and take the action described for that message.

System Programmer Response: None.

ASFP344I BATCHPIPES CLOSE FAILURE. AN ABEND OCCURRED UNDER THE SUBSYSTEM WHEN PROCESSING THE REQUEST: ABEND=*abendcode* RSN=*rsncode* JOB=*jobname* STEP=*jobstep* DD=*ddname* SUBSYS=*ssname*

Explanation: CLOSE processing for the pipe data set defined by the DD statement, *ddname*, failed because an abnormal end occurred when the BatchPipes subsystem address space was processing the request.

In the message text:

abendcode
is the abend completion code.

rsncode
is the abend reason code, or zero if there is no reason code associated with the abend.

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being closed.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSOC

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and will need to be rescheduled, have the operator cancel these jobs. Then notify your system programmer.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: Obtain the SYSLOG and the SVC Dump associated with the indicated abend and determine the cause of the problem. If the failure is due to an installation error, correct the problem and then notify the application programmer responsible for the pipeline so that all affected jobs can be rescheduled. Otherwise, contact the IBM Support Center.

ASFP345I BATCHPIPES CATASTROPHIC FAILURE. NOT ABLE TO PROCESS PIPES. JOB=*jobname* STEP=*jobstep* DD=*ddname* SUBSYS=*ssname*

Explanation: Processing for the pipe data set defined by the DD statement, *ddname*, failed because the BatchPipes subsystem *ssnm* is no longer able to perform allocation processing. A catastrophic error condition occurred under the BatchPipes subsystem address space that prevents it from processing any further requests. The subsystem has either lost a critical service required for processing or has encountered a critical error with the BatchPipe coupling facility structure.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being allocated.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSAA ASFPMSAU

System Action: The job is abnormally terminated.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and will need to be rescheduled, have the operator cancel these jobs. Then notify your system programmer.

Once the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed as well as all other jobs in the pipeline that were affected by the error.

System Programmer Response: Examine the SYSLOG at the time of the job failure. If the BatchPipes subsystem on the system where the job was running lost a critical service task due to an abnormal termination, obtain the SVC dump associated with abend and determine the cause of the problem. On the other hand, if the SYSLOG at the time of the ,'job failure shows that message ASFP429E was issued indicating a BatchPipes coupling facility structure failure or connectivity problem occurred, determine the cause of the problem. If the problem is due to an installation error, correct the problem and then notify the application programmer responsible for the the pipeline so that all affected jobs may be rescheduled. Otherwise, contact the IBM Support Center.

ASFP349I BATCHPIPES FAILURE. TERMSYNC IS MUTUALLY EXCLUSIVE WITH EROPT=SKP OR EROPT=ACC. JOB=*jobname* STEP=*jobstep* DD=*ddname* SUBSYS=*ssname*

Explanation: Processing for the pipe data set that is defined by the DD statement, *ddname*, failed because either EROPT=SKP, EROPT=ACC, ERRPROP=DUMMY, or ERRPROP=CONT was specified on a pipe that also specified TermSync. TermSync is mutually exclusive with EROPT=SKP or EROPT=ACC and ERRPROP=DUMMY or ERRPROP=CONT

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being processed at the time of the failure.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSOO

System Action: Open processing fails for the pipe.

Operator Response: None.

User Response: None.

Application Programmer Response: Correct the problem by removing EROPT=SKP, EROPT=ACC, ERRPROP=DUMMY, or ERRPROP=CONT or TermSync from the pipe DD statement.

When the problem is corrected, do data set cleanup that may be required if any to rerun the jobs. Rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: None.

ASFP350I BATCHPIPES CLOSE FAILURE: ABEND=*abendcode* RSN=*rsncode* JOB=*jobname* STEP=*jobstep* DD=*ddname* SUBSYS=*ssname*

Explanation: An unexpected abnormal end occurred during BatchPipes CLOSE processing for the pipe data set defined by the DD statement, *ddname*.

In the message text:

abendcode
is the abend completion code.

rsncode
is the abend reason code, or zero if there is no reason code associated with the abend.

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being closed.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSOC

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and need to be rescheduled, have the operator cancel these jobs. Then determine the cause of the failure. If the failure is due to an application error correct the problem. Otherwise, notify your system programmer.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: Obtain the SYSLOG and the SVC Dump associated with the indicated abend and determine the cause of the problem. If the failure is due to an installation error, correct the problem and then notify the application programmer responsible for the pipeline so that all affected jobs can be rescheduled. Otherwise, contact the IBM Support Center.

ASFP351I BATCHPIPES OPEN FAILURE. UNSUPPORTED ACCESS METHOD. JOB=*jobname* STEP=*jobstep* DD=*ddname* SUBSYS=*ssname* ACBSTYP=*acbstyp*

Explanation: Open processing for the pipe data set defined by the DD statement, *ddname*, failed because the type of access method indicated by the ACB being opened is not one of the sequential access methods supported by BatchPipes. The ACBSTYP field indicates VSAM, VTAM, or some other unsupported type.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being opened.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

acbstyp
is a 1 byte field in the ACB that qualifies the type of access method to be used.

Source: BATCHPIPES

Detecting Module:
ASFPMSOO

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and need to be rescheduled, have the operator cancel these jobs. Then correct the application to use an access method that is supported by BatchPipes (i.e., BSAM or QSAM).

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: None.

ASFP352I BATCHPIPES OPEN FAILURE: ABEND=*abendcode* **RSN=***rsncode* **JOB=***jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname*

Explanation: An unexpected abnormal end occurred during BatchPipes open processing for the pipe data set defined by the DD statement, *ddname*.

In the message text:

abendcode
is the abend completion code.

rsncode
is the abend reason code, or zero if there is no reason code associated with the abend.

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being opened.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSOO

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and need to be rescheduled, have the operator cancel these jobs. Then determine the cause of the failure. If the failure is due to an application error correct the problem. Otherwise, notify your system programmer.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: Obtain the SYSLOG and the SVC Dump associated with the indicated abend and determine the cause of the problem. If the failure is due to an installation error, correct the problem and then notify the application programmer responsible for the pipeline so that all affected jobs can be rescheduled. Otherwise, contact the IBM Support Center.

ASFP353I BATCHPIPES OPEN FAILURE. DATA SET ALREADY OPEN. JOB=*jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname*

Explanation: Open processing for the pipe data set defined by the DD statement, *ddname*, failed because the data set was already open under the jobstep.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being opened.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSOO

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and need to be rescheduled, have the operator cancel these jobs. Then correct the application that was being executed by the job so it does not attempt to open a pipe data set that is already open.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: None.

ASFP354I BATCHPIPES CLOSE FAILURE. SUBSYSTEM NOT ACTIVE. JOB=*jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname*

Explanation: Close processing for the pipe data set defined by the DD statement, *ddname*, failed because the BatchPipes subsystem that was managing the pipe is no longer active.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being closed.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSOC

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure and do any data set cleanup that may be required to rerun the jobs. Then contact your operator or system programmer to ensure the BatchPipes subsystem has been restarted.

When the BatchPipes subsystem has been restarted, rerun the job that failed and all other jobs in the pipeline that were affected by the failure.

System Programmer Response: None.

ASFP355I BATCHPIPES FAILURE. OPENSYNC VALUES DO NOT MATCH PIPE. JOB=*jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname*

Explanation: Processing for the pipe data set defined by the DD statement, *ddname*, failed because the OPENSYNC subsystem parameter values specified on the DD statement do not match the OPENSYNC subsystem parameter values of the existing pipe.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being processed.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSOO ASFPMXAA

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine whether this job should connect to the pipe data set identified by the DSN= and SUBSYS= keywords. If not, correct the DSN= or SUBSYS= keyword(s) to identify the correct pipe data set and BatchPipes subsystem. Otherwise determine the appropriate OPENSYNC subsystem parameter values and update the job(s) which incorrectly specify the values.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: None.

ASFP356I BATCHPIPES OPEN FAILURE. SUBSYSTEM NOT ACTIVE. JOB=*jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname*

Explanation: Open processing for the pipe data set defined by the DD statement, *ddname*, failed because the BatchPipes subsystem that was managing the pipe is no longer active.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being opened.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSOC

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure and do any data set cleanup that may be required to rerun the jobs. Then contact your operator or system programmer to ensure the BatchPipes subsystem has been restarted.

When the BatchPipes subsystem has been restarted, rerun the job that failed and all other jobs in the pipeline that were affected by the failure.

System Programmer Response: None.

ASFP357I BATCHPIPES OPEN FAILURE. INOUT OR OUTIN SPECIFIED. JOB=*jobname* STEP=*jobstep* DD=*ddname* SUBSYS=*ssname*

Explanation: Open processing for the pipe data set defined by the DD statement, *ddname*, failed because the pipe data set was being opened for INOUT or OUTIN. Neither of these open options are supported for a pipe data set.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being opened.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSOC

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and need to be rescheduled, have the operator cancel these jobs. Then correct the error by specifying either LABEL=(,,,IN) or LABEL=(,,,OUT) on the DD statement for the pipe data set to force it to be opened for either input only or output only depending on the application.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: None.

ASFP358I AUTHORITY CHECK FAILED: *jobname*, *jobstep*, *ddname*, *ssname*, *rname*

Explanation: Allocation for the pipe defined by the DD statement, *ddname*, failed due to insufficient access authority to the resource, *rname*. If the resource name is the name of the pipe data set, that was specified on the DD statement, then the user is not properly authorized to access the specified pipe data set. Otherwise, the problem is that the user is not authorized (through the FACILITY class) to use the specified subsystem.

In the message text:

jobname
is the name of the failing job.

jobstep
is the name of the failing job step.

ddname
is the DD name of the pipe data set that was being opened

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

rname
is the resource name that failed the authority check.

Source: BATCHPIPES

Detecting Module:
ASFPMSSC

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the failure. If any of them are still active in the system and need to be rescheduled, have the operator cancel these jobs. Then check to ensure that the correct data set name and subsystem name were specified on the DD statement for the pipe data set that failed authorization. If either one of these resource names is incorrect, specify the correct name. Otherwise, contact your system security administrator to obtain proper authorization to the resource.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: None.

ASFP359I BATCHPIPES FAILURE. CLOSESYNC MISMATCH ON PIPE. JOB=*jobname* STEP=*jobstep* DD=*ddname* SUBSYS=*ssname*

Explanation: Pprocessing for the pipe data set defined by the DD statement, *ddname*, failed because either this connection did specify the CLOSESYNC subsystem parameter and the existing pipe did not specify CLOSESYNC, or this connection did not specify the CLOSESYNC subsystem parameter and the existing pipe did specify CLOSESYNC.

In the message text:

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being processed.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSOO/ASFPMXAA

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine whether this pipe should have CLOSESYNC specified and update all jobs connecting to the pipe to specify or not specify the subsystem parameter. It could also be that the pipename or subsystem name specified for this DD statement is not correct. If this is the case, correct the subsystem name or pipename to the appropriate value.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: None.

ASFP360I BATCHPIPES I/O READ ERROR: JOB=*jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname* **RSN=***rsncode* **RPLFDBWD=***rplfdbwd*

Explanation: In the job where this message appears, an I/O error condition occurred during the processing of a READ or GET request for the pipe defined by the DD statement, *ddname*. Because of the error, no record was returned for the READ or GET request.

In the message text:

jobname
is the name of the job that encountered the error.

jobstep
is the name of the current jobstep.

ddname
is the DD name of the pipe affected by the error.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

rsncode
is a hexadecimal error reason code as follows:

D7621030 attempted to read after close. The connection to the pipe for this DD and all partner connections were closed.

D7622030 attempted to read after close. The connection to the pipe for this DD was closed.

D7625890 attempted to read from the pipe before the completion of a previous read request which is still waiting to be satisfied.

D762D0D0 an unexpected abnormal end occurred during BatchPipes processing of a READ or GET request.

E25E4030 an unexpected abnormal end occurred during BatchPipes processing of a READ or GET request.

rplfdbwd
is the RPLFDBWD field (i.e., feedback word) from the RPL that was used by BatchPipes input processing to process the READ or GET request.

Source: BATCHPIPES

Detecting Module:
ASFPMSGC

System Action: If a SYNAD routine was specified for the data set at OPEN time, the error condition is passed to the SYNAD routine. If no SYNAD routine was specified, but an EROPT DD parameter or ERRPROP Subsys keyword was specified, then processing proceeds according to the particular error option specified. If neither a SYNAD routine or EROPT parameter was specified, the job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the error. If any of them are still active in the system and need to be rescheduled, have the operator cancel these jobs. Then determine the cause of the error. If the error was due to an application problem, correct the problem. Otherwise, notify your system programmer.

When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: Obtain the SYSLOG and any SVC Dump that may be associated with the error Then determine the cause of the error. If the error is due to an installation problem, correct the problem and then notify the application programmer responsible for the pipeline so that all affected jobs can be rescheduled. Otherwise, contact the IBM Support Center.

ASFP361I BATCHPIPES OPEN FAILURE. FITTING INITIALIZATION ERROR.
JOB=*jobname* **STEP=***jobstep*
DD=*ddname* **SUBSYS=***ssname*
RSN=*rsncode*

Explanation: Open processing for the pipe data set defined by the DD statement, *ddname*, failed because the an error occurred while initializing the fitting that was specified.

In the message text:

jobname
is the name of the job that encountered the error.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being opened.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

rsncode
is a hexadecimal error reason code.

Source: BATCHPIPES

Detecting Module:
ASFPMSOO

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: An accompanying message was issued describing why the fitting initialization failed. Correct the problem and rerun the job.

System Programmer Response: None.

ASFP362I BATCHPIPES CLOSE WARNING: DATA MAY BE
INCOMPLETE DUE TO *reason-text*
JOB=*jobname* **JOBSTEP=***jobstep*
DD=*ddname* **SUBSYS=***ssname*

Explanation: Close processing for the pipe data set defined by the DD statement, *ddname*, determined that it is possible for the data flowing through the pipe data set to be incomplete because the job was either cancelled or a failure occurred during PUT processing.

In the message text:

reason-text
One of the following:

CANCEL.
Because a CANCEL was issued, output to the pipe may have terminated prematurely.

A FAILURE DURING PUT PROCESSING.
A failure during PUT processing prevented BatchPipes from writing all records to the pipe successfully.

jobname
is the name of the job that was writing to the pipe.

jobstep
is the name of the jobstep.

ddname
is the name of the DD statement for the pipe data set that is receiving this warning.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSOC

System Action: Processing continues.

Operator Response: None.

User Response: None.

Application Programmer Response: Ensure that the data is complete before allowing subsequent processing to occur. It is possible that data could not be written to the pipe once cancel processing was started or once the PUT failure occurred.

System Programmer Response: None.

ASFP363I BATCHPIPES CLOSE ERROR: EOF NOT RECEIVED BY READER
JOB=*jobname* **STEP=***jobstep*
DD=*ddname* **SUBSYS=***ssname*

Explanation: Reader processing for the pipe data set defined by the DD statement, *ddname*, requested a CLOSE before receiving an EOF indication and the EOFREQUIRED option was set to 'YES'.

In the message text:

jobname
is the name of the job that contains the DD statement.

jobstep
is the name of the step containing the DD statement.

ddname
is the name of the DD statement for the pipe data set that is receiving this warning.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSOC

System Action: The reader job is ABENDed with a x'BC6', Reason Code x'E2D64050'.

Operator Response: None.

User Response: None.

Application Programmer Response: Ensure that the data is complete before allowing subsequent processing to occur.

System Programmer Response: None.

ASFP364E BATCHPIPES SUBSYSTEM *ssname* **HAS DETECTED THAT JOBSTEP** *jobstep* **HAS ENDED WITH A COMPCODE OF** *ccvalue*. **THIS VALUE IS GREATER THAN OR EQUAL TO THE TERMSYNC VALUE (***tsvalue***) SPECIFIED ON DD** *ddname*. **THE STEP IS ENDED.**

Explanation: The jobstep ended with a step completion code that will cause BatchPipes to abend the step.

In the message text:

ssname
is the name of the BatchPipes subsystem detecting the condition.

jobstep
is the name of the jobstep that is ending.

ccvalue
is the completion code of the step that is ending.

tsvalue
is the value specified for TERMSYNC.

ddname
is the name on the DD statement whose condition was exceeded.

Source: BATCHPIPES

Detecting Module:
ASFPMRR4

System Action: The job is ABENDed with a x'BC6', Reason Code x'E68710E4'.

Operator Response: None.

User Response: Determine the cause for the completion code and correct the problem. Then rerun the affected job(s).

Application Programmer Response: None.

System Programmer Response: None.

ASFP365I BATCHPIPES ERROR PROPAGATED: JOB=*jobname* **JOBSTEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname* **ERRORJOB=***ejobname* **ERRORSYS=***esysname* **ERROR-REASON: JOB** *ejobname* **IN PIPELINE** *reason-text***. JOB** *jobname* **WILL BE CANCELLED.**

Explanation: An error has been propagated to the job identified by JOB=*jobname*.The error propagation occurred because another job in the pipeline was either canceled or abnormally terminated. The job that caused the error propagation is identified by the ERRORJOB= text.

In the message text:

jobname
is the name of the job receiving the error propagation.

jobstep
is the name of the current jobstep in the job receiving the error propagation.

ddname
is the name of the DD statement for the pipe data set that this error propagation is associated with.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

ejobname
is the name of the job that encountered the error that caused the propagation.

esysname
is the name of the system on which the job encountered the error that caused the propagation.

reason-text
One of the following:

WAS CANCELED
indicates the error propagation occurred because a job in the pipeline identified by ERRORJOB= was canceled.

ABNORMALLY TERMINATED
indicates the error propagation occurred because a job in the pipeline identified by ERRORJOB= abnormally terminated.

Source: BATCHPIPES

Detecting Module:
ASFPMSRR

System Action: The job receiving the error propagation will be cancelled. All other jobs that are on the pipeline will also receive an error propagation and be abnormally terminated via cancel.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine what caused the job identified by ERRORJOB= to fail and correct the problem. Then do any data set cleanup that may be required and re-execute the stream of BatchPipes jobs that were affected by the failure.

System Programmer Response: None.

ASFP366E BATCHPIPES FAILURE. TERMSYNC SPECIFIED: JOBSTEP=*jobstep* DD=*ddname* SUBSYS=*ssname*

Explanation: Allocation processing for the pipe data set that was defined by the DD statement *ddname* failed because TermSync was specified on the DD statement for a non-TermSync pipeline.

In the message text:

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being allocated at the time of the failure.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSAA

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine which jobs in the pipeline affected by the failure. '

If any jobs are still active in the system and need to be rescheduled, have the operator cancel these jobs.

Remove *TermSync* from the SUBSYS= parameter.

When the problem is corrected, do any data set cleanup that might be necessary to rerun the jobs. When the clean-up is complete, rerun the failing job and any other jobs in the pipeline that were affected by the failure.

System Programmer Response: None.

ASFP367E BATCHPIPES FAILURE. TERMSYNC NOT SPECIFIED: JOBSTEP=*jobstep* DD=*ddname* SUBSYS=*ssname*

Explanation: Allocation processing for the pipe data set defined by the DD statement *ddname* failed because TermSync was not specified on the DD statement, and TermSync was in effect for this pipeline.

In the message text:

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that the job was allocating at the time of the failure.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSAA

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine which jobs in the pipeline were affected by the 'failure.

If any jobs are still active in the system and need to be rescheduled, have the operator cancel the jobs.

Correct the DD statement for the pipe data set so it specifies TERMSYNC.

When the problem is corrected, do any data set cleanup that might be necessary to rerun the jobs. When the clean-up is completed, rerun the job that failed, and any other jobs in the pipeline that were affected by the failure.

System Programmer Response: None.

ASFP368E BATCHPIPES FAILURE. ALLOCSYNC VALUE MISMATCH: JOBSTEP=*jobstep* DD=*ddname* SUBSYS=*ssname*

Explanation: Allocation processing for the pipe data set defined by the DD statement *ddname* failed because the AllocSync specification did not match the pipe data set.

In the message text:

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being allocated at the time of the failure.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSAA

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine which jobs in the pipeline were affected by the failure.

If any jobs are still active in the system and need to be rescheduled, have the operator cancel these jobs.

Correct the DD statement for the pipe data set so it specifies the correct AllocSync value.

When the problem is corrected, do any data set cleanup that might be necessary to rerun the jobs. When the clean-up is complete, rerun the job that failed, and any other jobs in the pipeline that were affected by the failure.

System Programmer Response: None.

ASFP369E BATCHPIPES FAILURE. *jobtype* **SPECIFIED** *keyword* **JOB=***jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname*

Explanation: Allocation processing for the pipe data set defined by the DD statement, *ddname*, failed because the *keyword* listed was used by the *jobtype* listed. AllocSync and TermSync cannot be used by TSO/E users or system address spaces.

In the message text:

jobtype
is the type of job that was attempting to allocate the pipe data set. The possible types are:

- TSO/E USER
- SYS ADDR SPACE

keyword
is the keyword that job attempted to use. The possible keywords are:

- ALLOCSYNC
- TERMSYNC

jobname
is the name of the job that failed.

jobstep
is the name of the jobstep that failed.

ddname
is the name of the DD statement for the pipe data set that was being allocated at the time of the failure.

ssname
is the name of the BatchPipes subsystem that was specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSAA

System Action: The job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine which jobs in the pipeline affected by the failure. '

If any jobs are still active in the system and need to be rescheduled, have the operator cancel these jobs.

Correct the DD statement for the pipe data set so it does not specify the *keyword* listed.

When the problem is corrected, do any data set cleanup that might be necessary to rerun the jobs. When the cleanup is completed, rerun the job that failed, and any other jobs in the pipeline that were affected by the failure.

System Programmer Response: None.

ASFP370I BATCHPIPES I/O WRITE ERROR: **JOB=***jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname* **RSN=***rsncode* **RPLFDBWD=***rplfdbwd*

Explanation: In the job where this message appears, an I/O error condition occurred during the processing of a WRITE or PUT request for the pipe defined by the DD statement, *ddname*. Because of the error, no record was written for the WRITE or PUT request.

In the message text:

jobname
is the name of the job that encountered the error.

jobstep
is the name of the current jobstep.

ddname
is the DD name of the pipe affected by the error.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

rsncode
is a hexadecimal error reason code as follows:

D76C1030 attempted to write after close. The connection to the pipe for this DD and all partner connections were closed.

D76C2030 attempted to write after close. The connection to the pipe for this DD was closed.

D76C3470 attempted to write to the pipe before the completion of a previous write request which is still waiting to be satisfied.

D76C5890 attempted to write to the pipe before the completion of a previous write request which is still waiting to be satisfied.

D76CD0D0 an unexpected abnormal end occurred during BatchPipes processing of a WRITE or PUT request.

E2664030 an unexpected abnormal end occurred during BatchPipes processing of a WRITE or PUT request.

rplfdbwd
is the RPLFDBWD field (i.e., feedback word) from the RPL that was used by BatchPipes input processing to process the READ or GET request.

Source: BATCHPIPES

Detecting Module:
ASFPMSPC

System Action: If a SYNAD routine was specified for the data set at OPEN time, the error condition is passed to the SYNAD routine. If no SYNAD routine was specified, but an EROPT DD parameter or ERRPROP Subsys keyword was specified, then processing proceeds according to the particular error option specified. If neither a SYNAD routine or EROPT parameter was specified, the job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Determine all jobs in the pipeline affected by the error. If any of them are still active in the system and need to be rescheduled, have the operator cancel these jobs. Then determine the cause of the error. If the error was due to an application problem, correct the problem. Otherwise, notify your system programmer. When the problem is corrected, do any data set cleanup that may be required to rerun the jobs. When this is completed, rerun the job that failed and all other jobs in the pipeline that were affected by the error.

System Programmer Response: Obtain the SYSLOG and any SVC Dump that may be associated with the error Then determine the cause of the error. If the error is due to an installation problem, correct the problem and then notify the application programmer responsible for the pipeline so that all affected jobs can be rescheduled. Otherwise, contact the IBM Support Center.

ASFP371I BATCHPIPES CLOSE ERROR: WRITE/READ BLOCK COUNT MISMATCH **JOB=***jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname*

Explanation: Reader processing for the pipe data set defined by the DD statement, *ddname*, requested a CLOSE but the combined sum of all the WRITE requests and all the READ requests for the whole pipe do not match and the user has requested EOFREQUIRED=YES (which enables this check to be made.

In the message text:

jobname
is the name of the job that contains the DD statement.

jobstep
is the name of the step containing the DD statement.

ddname
is the name of the DD statement for the pipe data set that is receiving this warning.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSOC

System Action: The reader job is ABENDed with a x'BC6', Reason Code x'E7930440'.

Operator Response: None.

User Response: None.

Application Programmer Response: Ensure that the data is complete before allowing subsequent processing to occur.

System Programmer Response: None.

ASFP374I BATCHPIPES I/O ERROR PROPAGATED: JOB=*jobname* **STEP=***curstep* **DD=***ddname* **SUBSYS=***ssname* **PARTNER=(***jobid***,** *jobname2***,** *jobstep***,** *sysname***,** *errtype***)**

Explanation: In the job where this message appears, the pipe data set defined by the DD statement, *ddname*, received an I/O error that was propagated because of an error condition encountered by a partner job connected to the pipe data set. Either the partner job abnormally ended or the BatchPipes subsystem servicing the partner job failed. The *errtype* field in the message indicates which one of these conditions occurred. In either case, the partner job has been lost from the BatchPipes job stream that was being executed.

In the job which received this propagated error message, neither EROPT=SKP or EROPT=ACC DD parameter or ERRPROP=DUMMY or ERRPROP=CONT Subsystem keyword was specified for the pipe data set.

In the message text:

jobname
is the name of the job that received the propagated error.

curstep
is the name of the current jobstep in the job that received the propagated error.

ddname
is the DD name of the pipe data set affected by the error.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

jobid
is the job ID of the partner job that was connected to the pipe and encountered an error.

jobname2
is the name of the partner job that was connected to the pipe and encountered an error.

jobstep
is the name of the failing job step in the partner job that was connected to the pipe and encountered an error.

sysname
is the name of the system on which the partner job was executing.

errtype
One of the following:

JOB-ERR
indicates the error occurred because the partner job abnormally ended.

SUBSYS-ERR
indicates the error occurred because the BatchPipes subsystem that was servicing the partner job failed.

Source: BATCHPIPES

Detecting Module:
ASFPMSGC ASFPMSPC

System Action: The current input or output request completes successfully since the I/O error condition is really only a logical error that is triggered as the result of an error encountered by the partner job. However, because neither EROPT=SKP or EROPT=ACC ERRPROP=DUMMY or ERRPROP=CONT Subsystem keyword was specified, the propagated error condition is not ignored. Instead, the error condition is passed to the SYNAD routine associated with the data set if one was specified. If no SYNAD routine was specified, the job is abnormally ended.

Operator Response: None.

User Response: None.

Application Programmer Response: If the partner job failed (i.e., JOB-ERR), determine what caused it to fail and correct the problem. Then do any data set cleanup that may be required and re-execute the stream of BatchPipes jobs that were affected by the failure.

If the BatchPipes subsystem that was servicing the partner job failed (i.e., SUBSYS-ERR), check with your system programmer or operator to ensure that the BatchPipes subsystems required by your job stream are available (i.e., active) on the systems where your jobs may execute. Then do any data set cleanup that may be required and re-execute the stream of BatchPipes jobs that were affected by the failure.

System Programmer Response: None.

ASFP375I BATCHPIPES I/O ERROR PROPAGATED: EROPT=SKP JOB=*jobname* **STEP=***curstep* **DD=***ddname* **SUBSYS=***ssname* **PARTNER=(***jobid***,** *jobname2***,** *jobstep***,** *sysname***,** *errtype***)**

Explanation: In the job where this message appears, the pipe data set defined by the DD statement, *ddname*, received an I/O error that was propagated because of an error condition encountered by a partner job connected to the pipe data set. Either the partner job abnormally ended or the BatchPipes subsystem servicing the partner job failed. The *errtype* field in the message indicates which one of these conditions occurred. In either case, the partner job has been lost from the BatchPipes job stream that was being executed.

In the job which received this propagated error message, EROPT=SKP was specified for the pipe data set.

In the message text:

jobname
is the name of the job that received the propagated error.

curstep
is the name of the current jobstep in the job that received the propagated error.

ddname
is the DD name of the pipe data set affected by the error.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

jobid
is the job ID of the partner job that was connected to the pipe and encountered an error.

jobname2
is the name of the partner job that was connected to the pipe and encountered an error.

jobstep
is the name of the failing job step in the partner job that was connected to the pipe and encountered an error.

sysname
is the name of the system on which the partner job was executing.

errtype
One of the following:

JOB-ERR
indicates the error occurred because the partner job abnormally ended.

SUBSYS-ERR
indicates the error occurred because the BatchPipes subsystem that was servicing the partner job failed.

Source: BATCHPIPES

Detecting Module:
ASFPMSGC ASFPMSPC

System Action: The current input or output request completes successfully since the I/O error condition is really only a logical error that is triggered as the result of an error encountered by the partner job. Also, since EROPT=SKP was specified, the propagated error is ignored and the job is allowed to continue processing against the pipe. If the job is a reader job, it will be allowed to read from the pipe if there is still data in the pipe or there is at least one other partner job writing to the pipe. If the job is a writer job, it will be allowed to continue to write to the pipe. However, if there is no longer a reader job reading from the pipe, then the writer job will wait when the pipe becomes full.

Operator Response: None.

User Response: None.

Application Programmer Response: If the loss of the partner job caused the processing results of your job to be incomplete, do any data set cleanup that may be required and re-execute the stream of BatchPipes jobs that were affected by the failure.

System Programmer Response: None.

ASFP376I BATCHPIPES I/O ERROR PROPAGATED: EROPT=ACC JOB=*jobname* **STEP=***curstep* **DD=***ddname* **SUBSYS=***ssname* **PARTNER=(***jobid***,** *jobname***,** *jobstep***,** *sysname***,** *errtype***)**

Explanation: In the job where this message appears, the pipe data set defined by the DD statement, *ddname*, received an I/O error that was propagated because of an error condition encountered by a partner job connected to the pipe data set. Either the partner job abnormally ended or the BatchPipes subsystem servicing the partner job failed. The *errtype* field in the message indicates which one of these conditions occurred. In either case, the partner job has been lost from the BatchPipes job stream that was being executed.

In the job which received this propagated error message, EROPT=ACC was specified for the pipe data set.

In the message text:

jobname
is the name of the job that received the propagated error.

curstep
is the name of the current jobstep in the job that received the propagated error.

ddname
is the DD name of the pipe data set affected by the error.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

jobid
is the job ID of the partner job that was connected to the pipe and encountered an error.

jobstep
is the name of the failing job step in the partner job that was connected to the pipe and encountered an error.

sysname
is the name of the system on which the partner job was executing.

errtype
One of the following:

JOB-ERR
indicates the error occurred because the partner job abnormally ended.

SUBSYS-ERR
indicates the error occurred because the BatchPipes subsystem that was servicing the partner job failed.

Source: BATCHPIPES

Detecting Module:
ASFPMSGC ASFPMSPC

System Action: The current input or output request completes successfully since the I/O error condition is really only a logical error that is triggered as the result of an error encountered by the partner job. Also, since EROPT=ACC was specified, the propagated error is ignored and the job is allowed to continue processing against the pipe. If the job is a reader job, it will be allowed to read from the pipe if there is still data in the pipe or there is at least one other partner job writing to the pipe. If the job is a writer job, it will be allowed to continue to write to the pipe. However, if there is no longer a reader job reading from the pipe, then the writer job will wait when the pipe becomes full.

Operator Response: None.

User Response: None.

Application Programmer Response: If the loss of the partner job caused the processing results of your job to be incomplete, do any data set cleanup that may be required and re-execute the stream of BatchPipes jobs that were affected by the failure.

System Programmer Response: None.

ASFP377I BATCHPIPES WARNING. {BUFNO|PIPEDEPTH}(*value*) GREATER THAN MAX ALLOWED (*maxvalue*) - OPTION SET TO MAX. JOB=*jobname* STEP=*jobstep* DD=*ddname* SUBSYS=*ssname*

Explanation: The BUFNO DCB value or the PipeDepth subsystem parameter value specified on the allocation request is greater than the allowed maximum that has been defined by the BatchPipes MAXBUFNO parmlib option setting. If PipeDepth was specified, BatchPipes check with the security product and determined that this job was not authorized to use a setting larger than the MAXBUFNO setting.

In the message text:

BUFNO
The BUFNO DCB option was specified.

PIPEDEPTH
The PIPEDEPTH subsystem keyword was specified.

value
is the value specified on the BUFNO DCB option, or on the PipeDepth subsystem option on the DD statement for the pipe data set.

maxvalue
is the maximum PipeDepth value allowed as specified by the MAXBUFNO parmlib setting.

jobname
is the name of the job that contains the DD statement.

jobstep
is the name of the jobstep that contains the DD statement.

ddname
is the name of the DD statement that defines the pipe.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSAA

System Action: The PipeDepth for the pipe data set is set to the maximum allowed value (*maxvalue*). Processing of the pipe data set continues.

Operator Response: None.

User Response: Determine if the maximum value, as reported in the message, is sufficient for this pipe dataset. If it is not, then contact the system programmer to request that the MAXBUFNO parmlib setting be increased for this BatchPipes subsystem, or that this job is authorized to specify a PipeDepth value larger than the maximum allowable value.

Application Programmer Response: None.

System Programmer Response: None.

ASFP378I BATCHPIPES WARNING. {BUFNO|PIPEDEPTH}(*value*) DOES NOT MATCH PIPE (*pipevalue*) - OPTION SET TO PIPE VALUE. JOB=*jobname* STEP=*jobstep* DD=*ddname* SUBSYS=*ssname*

Explanation: The BUFNO DCB value or the PipeDepth subsystem parameter value specified on the allocation request does not match what has been defined for the pipe (the value is established by the FIRST connection to the pipe).

In the message text:

BUFNO
The BUFNO DCB option was specified.

PIPEDEPTH
The PIPEDEPTH subsystem keyword was specified.

value
is the value specified on the BUFNO DCB option, or on the PipeDepth subsystem option on the DD statement for the pipe data set.

pipevalue
is the value for the parm that has been established for the pipe.

jobname
is the name of the job that contains the DD statement.

jobstep
is the name of the jobstep that contains the DD statement.

ddname
is the name of the DD statement that defines the pipe.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSOO

System Action: The PipeDepth/BUFNO parm for the pipe data set is set to the value defined for the pipe (*pipevalue*). Processing of the pipe data set continues.

Operator Response: None.

User Response: Determine if the pipe value, as reported in the message, is sufficient for this pipe dataset. If it is not, then check the other connections to this pipe and set the PipeDepth/BUFNO parm on those connections to the appropriate value for the pipe.

Application Programmer Response: None.

System Programmer Response: None.

ASFP390I BATCHPIPES SUBSYSTEM *ssname*: PRODID=*productid* PRODLVL=*productlvl* COMPID=*compid* JOB=*jobname* STEP=*jobstep* DD=*ddname* SUBSYS=*ssname*

DCB=(LRECL=*lrecl*,BLKSIZE=*blksize*,RECFM=*recfm*), SUBSYSMODE(*subsystem-mode*) *pipemodeoptions*

Explanation: This message reports the DCB characteristics and any special options in affect for the pipe data set defined by the DD statement, *ddname*.

In the message text:

ssname
is the name of the BatchPipes subsystem.

productid
is the BatchPipes product id.

productlvl
is the BatchPipes product level.

compid
is the BatchPipes component id.

jobname
is the name of the job that contains the DD statement.

jobstep
is the name of the jobstep that contains the DD statement.

ddname
is the name of the DD statement that defines the pipe.

lrecl
is the logical record length for the pipe data set.

blksize
is the blocksize for the pipe data set.

recfm
is the record format for the pipe data set.

BUFNO=
The BUFNO DCB option was specified.

PIPEDEPTH=
The PIPEDEPTH subsystem keyword was specified.

pipedepth
is the number of buffer for the pipe data set.

errrorprop-setting
One of the following:

EROPT=ABE
The EROPT DCB option was specified and ABE was selected.

EROPT=ACC
The EROPT DCB option was specified and ACC was selected.

EROPT=SKP
The EROPT DCB option was specified and SKP was selected.

ERRPROP=ABEND
The ERRPROP subsystem option was specified and ABEND was selected.

ERRPROP=CANCEL
The ERRPROP subsystem option was specified and CANCEL was selected.

ERRPROP=CONT
The ERRPROP subsystem option was specified and CONT was selected.

ERRPROP=DUMMY
The ERRPROP subsystem option was specified and DUMMY was selected.

subsystem-mode
One of the following:

LOCAL
indicates the subsystem is in local mode and can be used only for piping data between jobs executing on the same system.

XSYS
indicates the subsystem is in cross-system mode and can be used for piping data between jobs executing on the same system or on different systems which are part of the same pipeplex.

pipemode
One of the following:

indicates the subsystem was in local mode.

MODE(LOCAL)
indicates the pipe was in local mode when the pipe was opened.

MODE(XSYS)
indicates the pipe was in cross-system mode when the pipe was opened.

options
lists the special options that are in effect for this pipe data set. The list of options is:

FITTING indicates that a PipeWorks Fitting has been requested.

Source: BATCHPIPES

Detecting Module:
ASFPMSOO

System Action: None.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP391I JOB=*jobname* **STEP=***jobstep* **DD=***ddname*
SUBSYS=*ssname*
BLKS-IN=*numblks*
WAITS=(*busywaits***,***emptywaits***)***pipemode*
OPENWAIT=*opentime*
IOWTIME=*waittime* **CLOSEWAIT=***waittime*

Explanation: This message provides statistical data related to input processing that was performed on the pipe data set defined by the DD statement, *ddname*.

In the message text:

jobname
is the name of the job the pipe information is related to.

jobstep
is the name of the jobstep that contains the DD statement.

ddname
is the name of the DD statement that defines the pipe.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

numblks
is the number of blocks read from the pipe.

busywaits
is the total number of times BatchPipes input processing had to wait to read a block of data from the pipe because of a busy condition.

emptywaits
is the total number of times BatchPipes input processing had to wait to read a block of data from the pipe because the pipe was empty.

pipemode
One of the following:

indicates the subsystem was in local mode.

MODE(LOCAL)
indicates the pipe was in local mode when the pipe was closed.

MODE(XSYS)
indicates the pipe was in cross-system mode when the pipe was closed.

opentime
is the time that was spent waiting for a connection at open time.

waittime
is the total amount of time BatchPipes processing had to wait to read a block of data from the pipe because the pipe was empty.

Source: BATCHPIPES

Detecting Module:
ASFPMSAU ASFPMSOC

System Action: None.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP392I **JOB=***jobname* **STEP=***jobstep* **DD=***ddname*
SUBSYS=*ssname*
BLKS-OUT=*numblks*
WAITS=(*busywaits*,*fullwaits***)***pipemode*
OPENWAIT=*opentime*
IOWTIME=*waittime* **CLOSEWAIT=***waittime*

Explanation: This message provides statistical data related to output processing that was performed on the pipe data set defined by the DD statement, *ddname*.

In the message text:

jobname
is the name of the job the pipe information is related to.

jobstep
is the name of the jobstep that contains the DD statement.

ddname
is the name of the DD statement that defines the pipe.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

numblks
is the number of blocks written to the pipe.

busywaits
is the total number of times BatchPipes output processing had to wait to write a block of data to the pipe because of a busy condition.

fullwaits
is the total number of times BatchPipes output processing had to wait to write a block of data to the pipe because the pipe was full.

pipemode
One of the following:

indicates the subsystem was in local mode.

MODE(LOCAL)
indicates the pipe was in local mode when the pipe was closed.

MODE(XSYS)
indicates the pipe was in cross-system mode when the pipe was closed.

opentime
is the time that was spent waiting for a connection at open time.

waittime
is the total amount of time BatchPipes output processing had to wait to write a block of data to the pipe because the pipe was full.

Source: BATCHPIPES

Detecting Module:
ASFPMSAU ASFPMSOC

System Action: None.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP393I **JOB=***job* **STEP=***jobstep* **DD=***ddname*
SUBSYS=*ssname*
{READER|WRITER} PROCESSING ENDED WHILE WAITING FOR PARTNER TO OPEN.
OPENWAIT=*opentime*

Explanation: This message is issued when open processing is ended while a job is waiting for a partner on the specified pipe. The *OPENWAIT* time shows the amount of time spent waiting before the end occurred.

In the message text:

job
is the name of the job containing the DD statement.

jobstep
is the name of the jobstep containing the DD statement.

ddname
is the name of the DD statement defining the pipe.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

READER
This job is opening the pipe data set in read mode.

WRITER
This job is opening the pipe data set in write mode.

opentime
is the time spent waiting for a connection at open time.

Source: BATCHPIPES

Detecting Module:
ASFPMSRT

System Action: None.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP394I BATCHPIPES READER JOB WAITING FOR OPEN.
JOB=*jobname* **STEP=***jobstep*
DD=*ddname* **SUBSYS=***ssname*
PIPE=*pipename*

Explanation: This message is issued to notify the user that this READER connection is waiting for other connections on the pipe to be opened before it can completed its open processing on *pipename*.

This WAIT may be caused by a missing WRITER connection in the simplest case, or by missing other READER connections if this connection was defined with OPENSYNC controls.

In the message text:

jobname
is the name of the job that is waiting for partners to open.

jobstep
is the name of the jobstep that contains the DD statement.

ddname
is the name of the DD statement that defines the pipe.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

pipename
is the name of the pipe that the connection is on.

Source: BATCHPIPES

Detecting Module:
ASFPMSOO

System Action: The job containing the connection is put into a WAIT by the subsystem until the OPEN criteria has been met. If this condition persists then the WAITING FOR OPEN inactivity threshold may be exceeded and message ASFP414E will be issued.

Operator Response: None.

User Response: Use the BatchPipes ISPF monitor to determine the status of the other connections for this pipe and look for failed jobs or for jobs that are still awaiting scheduling.

Application Programmer Response: If this WAIT condition is not satisfied in a timely manner then check into the scheduling policy for the set of jobs that this job is a part of.

System Programmer Response: None.

ASFP395I BATCHPIPES WRITER JOB WAITING FOR OPEN.
JOB=*jobname* **STEP=***jobstep*
DD=*ddname* **SUBSYS=***ssname*
PIPE=*pipename*

Explanation: This message is issued to notify the user that this WRITER connection is waiting for other connections on the pipe to be opened before it can completed its open processing on *pipename*.

This WAIT may be caused by a missing READER connection in the simplest case, or by missing other WRITER connections if this connection was defined with OPENSYNC controls.

In the message text:

jobname
is the name of the job that is waiting for partners to open.

jobstep
is the name of the jobstep that contains the DD statement.

ddname
is the name of the DD statement that defines the pipe.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

pipename
is the name of the pipe that the connection is on.

Source: BATCHPIPES

Detecting Module:
ASFPMSOO

System Action: The job containing the connection is put into a WAIT by the subsystem until the OPEN criteria has been met. If this condition persists then the

WAITING FOR OPEN inactivity threshold may be exceeded and message ASFP413E will be issued.

Operator Response: None.

User Response: Use the BatchPipes ISPF monitor to determine the status of the other connections for this pipe and look for failed jobs or for jobs that are still awaiting scheduling.

Application Programmer Response: If this WAIT condition is not satisfied in a timely manner then check into the scheduling policy for the set of jobs that this job is a part of.

System Programmer Response: None.

ASFP396I ALL READERS CLOSED. ERC=DUMMY NOW IN EFFECT. JOB=*jobname* **STEP=***jobstep* **DD=***ddname* **SUBSYS=***ssname*

Explanation: This message is issued to notify the user that this pipe has no active reader connections. ERC=DUMMY was specified on the SUBSYS parameter, which indicates that once the pipe has no readers, future I/O to the pipe is avoided. Data will still flow into a fitting if one is specified on the DD statement.

In the message text:

jobname
is the name of the job using the pipe that has been dummied.

jobstep
is the name of the jobstep that contains the DD statement.

ddname
is the name of the DD statement that defines the pipe.

ssname
is the name of the BatchPipes subsystem specified on the DD statement.

Source: BATCHPIPES

Detecting Module:
ASFPMSPC

System Action: Future writes to the pipe are not performed. Return codes from the write operations are always successful. If a fitting was specified, the fitting will continue to receive all data.

Operator Response: None.

User Response: None. This message is informational.

Application Programmer Response: None. This message is informational.

System Programmer Response: None.

ASFP399I BATCHPIPES *ssname* **ERROR IN PARAMETER:** *pppppppp error-text*

Explanation: In the job that received this message, a parameter error was detected when processing the sub-parameters specified by SUBSYS= on a DD statement for a pipe data set.

In the message text:

ssname
is the name of the BatchPipes subsystem.

pppppppp
is the name of the parameter in error or blank.

error-text
One of the following:

UNDEFINED PARAMETER
The parameter is not a supported parameter.

')' WAS EXPECTED BUT 'x' WAS FOUND
Where a right parenthesis was expected, the character "x" was found.

'(' OR '=' WAS EXPECTED BUT 'x' WAS FOUND
Where a left parenthesis or equal sign was expected, the character "x" was found.

VALUE EXCEEDS nnnnnn
The parameter value was larger than the maximum value allowed (i.e., nnnnnnn).

VALUE IS LESS THAN nnnnnn
The parameter value was less than the minimum value allowed (i.e., nnnnnnn).

VALUE IS NOT NUMERIC
The parameter value contains a non-numeric character.

VALUE MUST BE SPECIFIED
No parameter value was specified.

VALUE MUST BE 'YES' OR 'NO'
The parameter value specified must be either YES of NO.

FIRST CHARACTER IS NUMERIC
The parameter which is a non-numeric parameter incorrectly begins with a numeric character.

SPECIFIED MORE THAN ONCE
The parameter is incorrectly specified more than once.

ALL CHARACTERS MUST BE ALPHANUMERIC OR NATIONAL
The parameter contains an invalid character that is not alphanumeric or national.

NAME HAS MORE THAN 8 CHARACTERS
The specified parameter name is longer than 8 characters.

VALUE HAS MORE THAN 8 CHARACTERS
The specified parameter value is longer than 8 characters.

FIRST CHARACTER MUST BE ALPHABETIC OR NATIONAL
The specified parameter value must begin with an alphabetic or national character.

VALUE SPECIFIED BUT NONE EXPECTED
A parameter value was specified, but none is allowed.

FITDD MUST BE SPECIFIED IF CMT IS SPECIFIED.
The CMT= parameter must only be used with the FITDD= parameter.

FITDD AND FIT ARE MUTUALLY EXCLUSIVE.
Only FITDD or FIT may be specified, but not both.

MAXIMUM LENGTH OF VALUE IS 2.
The specified parameter value must be 1 or two characters.

INCORRECT VALUE SPECIFIED.
The specified parameter value was not an acceptable value.

OPENSYNC REQUIRES W= AND/OR R= VALUES.
The specified keyword requires the specification of additional keywords and values.

OPENSYNC W= AND R= VALUES SUMMED MUST NOT EXCEED nnnnn
The combined values specified for the OPENSYNC controls is invalid.

Source: BATCHPIPES

Detecting Module:
ASFPMSPP

System Action: The job is ended.

Operator Response: None.

User Response: None.

Application Programmer Response: Correct the parameter in error and rerun the job.

System Programmer Response: None.

ASFP401E BATCHPIPES *ssname* **READER JOB** *jobname* **IDLE ON PIPE** *pname*

Explanation: The job, *jobname*, which has the pipe data set, *pname,* open for input has not performed a read operation against the pipe data set for a period of time that exceeds the BatchPipes IDLE inactivity threshold.

The job is not being delayed by BatchPipes processing. Instead, the delay must be because the job is busy in some other part of its own processing or because the job has been held up by the system.

When this condition occurs, the entire pipeline of jobs (which job *jobname* is a part) may ultimately be held up.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the reader job that is idle.

pname
is the name of the pipe data set.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: The job is allowed to continue to execute. However, because it is not reading data from the pipe data set, this may cause jobs that are writing to the pipe data set to wait when the pipe is full.

Operator Response: Notify your system programmer.

User Response: None.

Application Programmer Response: None.

System Programmer Response: This message may be occurring simply because the IDLE inactivity threshold was set too low. If so, use the BatchPipes SET command to re-specify the IDLE inactivity threshold. This message may also occur because IDLE was specified on the SUBSYS statement of the job(s) running under the BatchPipes subsystem. If so, this can only be changed in the JCL on the next invocation of the job(s). Otherwise, determine what is preventing the job from reading data and correct the problem.

ASFP402I BATCHPIPES *ssname* **READER JOB** *jobname* **NO LONGER IDLE ON PIPE** *pname*

Explanation: The job, *jobname*, which has the pipe data set, *pname*, open for input has now begun reading from the pipe data set. Before this message, message ASFP401E was issued.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the reader job that was idle.

pname
is the name of the pipe data set on which the reader job was idle.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: Action message ASFP401E which was previously issued to report that the job was idle too long is deleted.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP403E BATCHPIPES *ssname* READER JOB *jobname* WAITING FOR DATA ON PIPE *pname*

Explanation: The job, *jobname*, which has the pipe data set, *pname*, open for input has tried to read data from the pipe data set, but the read operation is waiting because the pipe is empty. This condition has now existed for a period of time that exceeds the BatchPipes WAIT inactivity threshold.

This problem will occur when there is no writer job writing to the pipe data set or the writer job is held up by the system. When this condition occurs, the entire pipeline of jobs (which job *jobname* is a part) may ultimately be held up.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe.

jobname
is the name of the reader job that is waiting.

pname
is the data set name of the pipe on which the reader job is waiting for data.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: The job is allowed to remain active on the system.

Operator Response: Notify your system programmer.

User Response: None.

Application Programmer Response: None.

System Programmer Response: This message may be occurring simply because the WAIT inactivity threshold was set too low. If so, use the BatchPipes SET command to re-specify the WAIT inactivity threshold. This message may also occur because WAIT was specified on the SUBSYS statement of the job(s) running under the BatchPipes subsystem. If so, this can only be changed in the JCL on the next invocation of the job(s). Otherwise, Issue the BatchPipes "cmdprefix STATUS PIPE=*pname*" command to display the job(s) using the pipe data set. If there is no writer job, then either immediately schedule the appropriate writer job(s) for execution or if necessary, have the operator end all jobs that comprise the pipeline that is being held up and reschedule the entire pipeline of jobs again. On the other hand, if there is at least one writer job, determine what is preventing the writer job(s) from writing to the pipe data set and correct the problem.

ASFP404I BATCHPIPES *ssname* READER JOB *jobname* NO LONGER WAITING FOR DATA ON PIPE *pname*

Explanation: The job, *jobname*, which has the pipe data set, *pname*, open for input is now able to read data from the pipe. Before this message, message ASFP403E was issued.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the reader job that was waiting.

pname
is the name of the pipe data set on which the reader job was waiting for data.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: Action message ASFP403E which was previously issued to report that the job was held up waiting for data is deleted.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP405E BATCHPIPES *ssname* READER JOB *jobname* WAITING FOR OPEN ON PIPE *pname*

Explanation: The job, *jobname*, is attempting to open the pipe data set, *pname*, for input, but is held up waiting for an OPEN by a writer partner. This condition has now existed for a period that exceeds the BatchPipes WAITOPEN inactivity threshold.

When this condition occurs, the entire pipeline of jobs (which job *jobname* is a part) may ultimately be held up.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the reader job that is waiting.

pname
is the name of the pipe data set on which the reader job is waiting for open.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: The job is allowed to remain active on the system.

Operator Response: Notify the application programmer responsible for the job.

User Response: None.

Application Programmer Response: This message may be occurring simply because the WAITOPEN inactivity threshold was set too low. If so, notify your system programmer. This message may also occur because WAITOPEN was specified on the SUBSYS statement of the job(s) running under the BatchPipes subsystem. If so, this can only be changed in the JCL on the next invocation of the job(s). Otherwise, the wait condition may be caused by one of the following problems:

1. The pipe data set name specified on either the reader job or related writer job may be incorrect.
2. A writer job for the pipe data set may have been scheduled for execution and abnormally ended before the related reader job, *jobname*, was scheduled into execution and opened the pipe data set.
3. A writer job for the pipe set may accidentally never have been submitted for execution.
4. A writer job for the pipe data set is active on the system, but is being held up from opening the pipe data set for output.
5. In the case where the reader job is one of several reader jobs that read the same pipe data set, the writer job for the pipe data set and at least one of the reader jobs may have already executed and completed processing before the current reader job, *jobname*, was scheduled into execution and opened the pipe data set.

Determine which of the above problems is the cause and correct the problem.

System Programmer Response: If the WAITOPEN inactivity threshold is set too low, use the BatchPipes SET command to re-specify the WAITOPEN inactivity threshold, 'or specify a higher value on the JCL'SUBSYS statement of the job(s) running under the BatchPipes subsystem 'on the next invocation of the job(s).

ASFP406I BATCHPIPES *ssname* READER JOB *jobname* NO LONGER WAITING FOR OPEN ON PIPE *pname*

Explanation: The job, *jobname*, has completed opening the pipe data set, *pname*, for input and is now ready to begin reading data from the pipe data set. Before this message, message ASFP405E was issued.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the reader job that was waiting.

pname
is the name of the pipe data set on which the reader job was waiting for open.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: Action message ASFP405E which was previously issued to report that the job was held up waiting for open is deleted.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP407E BATCHPIPES *ssname* WRITER JOB *jobname* IDLE ON PIPE *pname*

Explanation: The job, *jobname*, which has the pipe data set, *pname*, open for output has not performed a write operation against the pipe data set for a period of time that exceeds the BatchPipes IDLE inactivity threshold.

The job is not being delayed by BatchPipes processing. Instead, the delay must be because the job is busy in some other part of its own processing or because the job has been held up by the system.

When this condition occurs, the entire pipeline of jobs (which job *jobname* is a part) may ultimately be held up.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the writer job that is idle.

pname
is the name of the pipe data set on which the writer job is idle.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: The job is allowed to continue to execute. However, because it is not writing data to the pipe data set, this may cause jobs that are reading from the pipe data set to wait when the pipe is empty.

Operator Response: Notify your system programmer.

User Response: None.

Application Programmer Response: None.

System Programmer Response: This message may be occurring simply because the IDLE inactivity threshold was set too low. If so, use the BatchPipes SET command to re-specify the IDLE inactivity threshold. This message may also occur because IDLE was specified on the SUBSYS statement of the job(s) running under the BatchPipes subsystem. If so, this can only be changed in the JCL on the next invocation of the job(s). Otherwise, determine what is preventing the job from writing data and correct the problem.

ASFP408I BATCHPIPES *ssname* WRITER JOB *jobname* NO LONGER IDLE ON PIPE *pname*

Explanation: The job, *jobname*, which has the pipe data set, *pname*, open for output has now begun writing to the pipe data set. Before this message, message ASFP407E was issued.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the writer job that was idle.

pname
is the name of the pipe data set on which the writer job was idle.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: Action message ASFP407E which was previously issued to report that the job was idle too long is deleted.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP409E BATCHPIPES *ssname* WRITER JOB *jobname* WAITING FOR DATA ON PIPE *pname*

Explanation: The job, *jobname*, which has the pipe data set, *pname*, open for output has tried to write data to the pipe data set, but the write operation is waiting because the pipe is full. This condition has now existed for a period of time that exceeds the BatchPipes WAIT inactivity threshold.

This problem will occur when there is no reader job reading from the pipe data set or the reader job is held up by the system. When this condition occurs, the entire pipeline of jobs (which job *jobname* is a part) may ultimately be held up.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe.

jobname
is the name of the writer job that is waiting.

pname
is the data set name of the pipe on which the writer job is waiting for data.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: The job is allowed to remain active on the system.

Operator Response: Notify your system programmer.

User Response: None.

Application Programmer Response: None.

System Programmer Response: This message may be occurring simply because the WAIT inactivity threshold was set too low. If so, use the BatchPipes SET command to re-specify the WAIT inactivity threshold. This message may also occur because WAIT was specified on the SUBSYS statement of the job(s) running under the BatchPipes subsystem. If so, this can only be changed in the JCL on the next invocation of the job(s). Otherwise, Issue the BatchPipes "cmdprefix STATUS PIPE=*pname*" command to display the job(s) using the pipe data set. If there is no reader job, then either immediately schedule the appropriate reader job(s) for execution or if necessary, have the operator end all jobs that comprise the pipeline that is being held up and reschedule the entire pipeline of jobs again. On the other hand, if there is at least one reader job, determine what is preventing the reader job(s) from reading from the pipe data set and correct the problem.

ASFP410I BATCHPIPES *ssname* **WRITER JOB** *jobname* **NO LONGER WAITING FOR DATA ON PIPE** *pname*

Explanation: The job, *jobname*, which has the pipe data set, *pname*, open for output is now able to write data to the pipe data set. Before this message, message ASFP409E was issued.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the writer job that was waiting.

pname
is the name of the pipe data set on which the writer job was waiting for data.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: Action message ASFP409E which was previously issued to report that the job was held up waiting to write to the pipe data set is deleted.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP411E BATCHPIPES *ssname* **WRITER JOB** *jobname* **WAITING FOR OPEN ON PIPE** *pname*

Explanation: The job, *jobname*, is attempting to open the pipe data set, *pname*, for output, but is held up waiting for an OPEN by a reader partner. This condition has now existed for a period that exceeds the BatchPipes WAITOPEN inactivity threshold.

When this condition occurs, the entire pipeline of jobs (which job *jobname* is a part) may ultimately be held up.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the writer job that is waiting.

pname
is the name of the pipe data set on which the writer job is waiting for open.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: The job is allowed to remain active on the system.

Operator Response: Notify the application programmer responsible for the job.

User Response: None.

Application Programmer Response: This message may be occurring simply because the WAITOPEN inactivity threshold was set too low. If so, notify your system programmer. This message may also occur because WAITOPEN was specified on the SUBSYS statement of the job(s) running under the BatchPipes subsystem. If so, this can only be changed in the JCL on the next invocation of the job(s). Otherwise, the wait condition may be caused by one of the following problems:

1. The pipe data set name specified on either the writer job or related reader job may be incorrect.
2. A reader job for the pipe data set may have been scheduled for execution and abnormally ended before the related writer job, *jobname*, was scheduled into execution and opened the pipe data set.
3. A reader job for the pipe data set may accidentally never have been submitted for execution.
4. A reader job for the pipe data set is active on the system, but is being held up from opening the pipe data set for input.

5. In the case where the writer job is one of several writer jobs that write to the same pipe data set, the reader job for the pipe data set and at least one of the writer jobs may have already executed and completed processing before the current writer job, *jobname*, was scheduled into execution and opened the pipe data set.

Determine which of the above problems is the cause and correct the problem.

System Programmer Response: If the WAITOPEN inactivity threshold is set too low, use the BatchPipes SET command to re-specify the WAITOPEN inactivity threshold, 'or specify a higher value on the JCL'SUBSYS statement of the job(s) running under the BatchPipes subsystem 'on the next invocation of the job(s).

ASFP412I BATCHPIPES *ssname* WRITER JOB *jobname* NO LONGER WAITING FOR OPEN ON PIPE *pname*

Explanation: The job, *jobname*, has completed opening the pipe data set, *pname*, for output and is now ready to begin writing data to the pipe data set. Before this message, message ASFP411E was issued.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the writer job that was waiting.

pname
is the name of the pipe data set on which the writer job was waiting for open.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: Action message ASFP411E which was previously issued to report that the job was held up waiting for open is deleted.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP413E BATCHPIPES *ssname* WRITER JOB *jobname* WAITING FOR ADDITIONAL OPEN CONNECTIONS PIPE=*pname* W=*numwrtrs* R=*numrdrs*

Explanation: The job, *jobname*, is waiting in open on the pipe data set, *pname*, for additional connections. This job specified that *W=numwrtrs* and *R=numrdrs* must open the pipe before open 'processing can continue. This job has been waiting in this condition for a period exceeding the BatchPipes WAITOPEN inactivity threshold. ' 'This problem will occur when not enough jobs are submitted 'to reach the specified number of connections, or when jobs 'using the specified pipe are delayed in the system.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the writer job that was waiting.

pname
is the name of the pipe data set on which the writer job was waiting for open.

numwrtrs
is the number of remaining writer connections that open the pipe.

numrdrs
is the number of remaining reader connections that open the pipe.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: The job is allowed to remain on the system.

Operator Response: Notify the appropriate system programmers.

User Response: None.

Application Programmer Response: If Jobs are missing in the application, submit 'them. You may need to consult with your system 'programmer to determine what jobs are missing, or 'you can use the BatchPipes ISPF monitor to determine 'what jobs are active and what jobs are missing. 'Determine if some of the jobs are delayed in the 'system and ensure that they are scheduled, or 'resubmit them if necessary.

System Programmer Response: This message may be occurring simply because the 'WAITOPEN inactivity threshold was set too low. If so, 'use the BatchPipes SET command to specify a new WAITOPEN 'inactivity threshold.'This message may also occur because WAITOPEN was specified on the SUBSYS statement of the job(s) running under the BatchPipes subsystem. If so, this can only be changed in the JCL on the next invocation of the job(s).

ASFP414E BATCHPIPES *ssname* **READER JOB** *jobname* **WAITING FOR ADDITIONAL OPEN CONNECTIONS PIPE=***pname* **W=***numwrtrs* **R=***numrdrs*

Explanation: The job, *jobname*, is waiting in open on the pipe data set, *pname*, for additional connections. This job specified that *W=numwrtrs* and *R=numrdrs* must open the pipe before open 'processing can continue. This job has been waiting in this condition for a period exceeding the BatchPipes WAITOPEN inactivity threshold. ' 'This problem will occur when not enough jobs are submitted 'to reach the specified number of connections, or when jobs 'using the specified pipe are delayed in the system.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the reader job that was waiting.

pname
is the name of the pipe data set on which the reader job was waiting for open.

numwrtrs
is the number of remaining writer connections that open the pipe.

numrdrs
is the number of remaining reader connections that open the pipe.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: The job is allowed to remain on the system.

Operator Response: Notify the appropriate system programmers.

User Response: None.

Application Programmer Response: If Jobs are missing in the application, submit 'them. You may need to consult with your system 'programmer to determine what jobs are missing, or 'you can use the BatchPipes ISPF monitor to determine 'what jobs are active and what jobs are missing. 'Determine if some of the jobs are delayed in the 'system and ensure that they are scheduled, or 'resubmit them if necessary.

System Programmer Response: This message may be occurring simply because the 'WAITOPEN inactivity threshold was set too low. If so, 'use the BatchPipes SET command to specify a new WAITOPEN 'inactivity threshold.'This message may also occur because WAITOPEN was specified on the SUBSYS statement of the job(s) running under the BatchPipes subsystem. If so, this can only be changed in the JCL on the next invocation of the job(s).

ASFP415E BATCHPIPES *ssname* **JOB** *jobname* **WAITING FOR SYNCHRONOUS CLOSURE ON PIPE** *pname*

Explanation: Job *jobname* has issued a close against pipe *pname*. The first job processed for pipe *pname* requested a synchronous closure of the pipe. All jobs processing this pipe must close the pipe before processing is allowed to continue. Some processes have not yet closed the pipe, and this job has been in this condition for a period that exceeds the BatchPipes WAITCLOSE inactivity threshold.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the job that was waiting.

pname
is the name of the pipe data set on which the job was waiting for additional connections to close the pipe.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: The job is allowed to remain on the system.

Operator Response: Notify the appropriate system programmers.

User Response: None.

Application Programmer Response: You can use the BatchPipes ISPF monitor to determine what jobs are active and their current status. Determine if some of the jobs are delayed in the system and ensure that they are progressing as expected.

System Programmer Response: This message may be occurring simply because the WAITCLOSE inactivity threshold was set too low. If so, use the BatchPipes SET command to specify the new WAITCLOSE inactivity threshold. This message may also occur because WAITCLOSE was specified on the SUBSYS statement of the job(s) running under the BatchPipes subsystem. If so, this can only be changed in the JCL on the next invocation of the job(s). Otherwise, issue the BatchPipes STATUS PIPE=*pname* command to see all of the jobs currently processing the pipe. Determine if a problem exists with the suite of jobs, and correct the problems so that the pipe will be closed.

ASFP417E BATCHPIPES *ssname* **READER JOB** *jobname* **WAITING FOR DATA ON PIPE** *pname* **WAITEOF CONDITION IS PRESENT.**

Explanation: The job, *jobname*, which has the pipe data set *pname* open for input has tried to read data from the pipe data set, but the read operation is waiting because the pipe is empty. This condition has now existed for a period of time that exceeds the BatchPipes *NOEOF* inactivity threshold. This problem occurs when all writers have closed the pipe and the last writer that closed the pipe specified the *NOEOF* subsystem parameter.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the reader job that was waiting.

pname
is the name of the pipe data set on which the job was waiting for additional connections to close the pipe.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: The job is allowed to remain on the system.

Operator Response: Notify the appropriate system programmers.

User Response: None.

Application Programmer Response: None.

System Programmer Response: This message may be occurring simply because the WAITEOF inactivity threshold was set too low. If so, use the BatchPipes SET command to specify the new WAITEOF inactivity threshold. This message may also occur because WAITEOF was specified on the SUBSYS statement of the job(s) running under the BatchPipes subsystem. If so, this can only be changed in the JCL on the next invocation of the job(s). Otherwise, issue the BatchPipes STATUS PIPE=*pname* command to see all of the jobs currently using the pipe. If jobs are still active, determine if a problem exists with the jobs and correct if possible. If the reader jobs need to be ended, you can issue the BatchPipes EOF subsystem command to present EOF to the readers of this pipe. If writer jobs are missing, submit the necessary jobs.

ASFP418I BATCHPIPES *ssname* **WRITER JOB** *jobname* **NO LONGER WAITING FOR ADDITIONAL OPENS ON PIPE** *pname*

Explanation: The job, *jobname*, has completed opening the pipe data set, *pname*, for output and is now ready to begin writing data to the pipe data set. Before this message, message ASFP413E was issued.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the writer job that was waiting.

pname
is the name of the pipe data set on which the writer job was waiting for additional opens.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: Action message ASFP413E which was previously issued to report that the job was held up waiting for additional opens is deleted.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP419I BATCHPIPES *ssname* **READER JOB** *jobname* **NO LONGER WAITING FOR ADDITIONAL OPENS ON PIPE** *pname*

Explanation: The job, *jobname*, has completed opening the pipe data set, *pname*, for input and is now ready to begin reading data from the pipe data set. Before this message, message ASFP414E was issued.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the reader job that was waiting.

pname
is the name of the pipe data set on which the writer job was waiting for additional opens.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: Action message ASFP414E which was previously issued to report that the job was held up waiting for additional opens is deleted.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP420I BATCHPIPES *ssname* **JOB** *jobname* **NO LONGER WAITING FOR SYNCHRONOUS CLOSURE ON PIPE** *pname*

Explanation: The job, *jobname*, has completed closing the pipe data set, *pname*. Before this message, message ASFP415E was issued.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the pipe job that was waiting.

pname
is the name of the pipe data set on which the job was waiting for additional close.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: Action message ASFP415E which was previously issued to report that the job was held up waiting for additional closes is deleted.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP422I BATCHPIPES *ssname* **JOB** *jobname* **NO LONGER WAITING FOR DATA ON PIPE** *pname*

Explanation: The job, *jobname*, has received data from the pipe data set, *pname* or the BatchPipes EOF command was issued for the pipe. Before this message, message ASFP417E was issued.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the reader job that was waiting.

pname
is the name of the pipe data set on which the job was waiting for data.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: Action message ASFP417E which was previously issued to report that the job was held up waiting for data is deleted.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP423I BATCHPIPES *ssname* **STRUCTURE REBUILD ON** *sysname*
IS UNABLE TO CONNECT TO THE NEW STRUCTURE.
IXLCONN ERROR RSNCODE = *connect-rsncode*

Explanation: When attempting to connect to a new coupling facility structure during structure rebuild processing, the BatchPipes subsystem *ssname* on system *sysname* received an unexpected error from the IXLCONN connect service and cannot continue with the rebuild process.

In the message text:

ssname
is the name of the BatchPipes subsystem.

sysname
is the name of the system on which the BatchPipes subsystem is executing.

connect-rsncode
is the error reason code returned by the IXLCONN connect request.

Source: BATCHPIPES

Detecting Module:
ASFPMXRC

System Action: A "rebuild stop" is initiated to stop the structure rebuild across the pipeplex.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP424I BATCHPIPES *ssname* STRUCTURE REBUILD ON *sysname* HAS LOST CONNECTIVITY TO THE NEW STRUCTURE.

Explanation: When rebuilding the BatchPipes coupling facility structure, the BatchPipes subsystem *ssname* on system *sysname* lost connectivty to the 'new coupling facility structure that was being created and cannot continue with the rebuild process.

In the message text:

ssname
is the name of the BatchPipes subsystem.

sysname
is the name of the system on which the BatchPipes subsystem is executing.

Source: BATCHPIPES

Detecting Module:
ASFPMXSR

System Action: A "rebuild stop" is initiated to stop the structure rebuild across the pipeplex.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP425E BATCHPIPES *ssname* REBUILD PROCESSING FAILED ON *sysname*. *reason-text* THE BATCHPIPES SUBSYSTEM IS BEING CANCELED.

Explanation: Because of the critical error condition indicated by the *reason-text*, the BatchPipes subsystem *ssname* on system *sysname* is unable to complete a structure rebuild or rebuild-stop request that is currently in process and is unable to resume normal cross-system pipe processing.

In the message text:

ssname
is the name of the BatchPipes subsystem.

sysname
is the name of the system on which the BatchPipes subsystem is executing.

reason-text
One of the following:

A STRUCTURE FAILURE OCCURRED ON THE OLD STRUCTURE.
either the old BatchPipes Coupling Facility Structure or the Coupling Facility on which it resides encountered a failure and is no longer accessible.

CONNECTIVITY TO THE OLD STRUCTURE HAS BEEN LOST.
connectivity to the old BatchPipes Coupling Facility structure has been lost. As a result, the structure is no longer accessible.

AN ABEND OCCURRED DURING REBUILD.
During structure rebuild processing on the indicated system, an abend occurred that prevented rebuild processing from completing.

Source: BATCHPIPES

Detecting Module:
ASFPMXSR

System Action: The BatchPipes subsystem on the indicated system is canceled. If there are any remaining BatchPipes subsystems in the pipeplex, rebuild processing will continue.

Operator Response: When the coupling facility problem that caused the failure has been fixed, restart the BatchPipes subsystem again.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP426I BATCHPIPES *ssname* STRUCTURE REBUILD INITIATED FOR PIPEPLEX *pipeplex-name*. CROSS-SYSTEM PIPING IS TEMPORARILY HALTED.

Explanation: As a result of a structure rebuild request, all cross-system pipe processing in the BatchPipes pipeplex, *plex-name*, has been stopped and rebuild processing has been initiated to rebuild the coupling facility structure used by the pipeplex. The coupling facility structure will be rebuilt on either the same coupling facility or another coupling facility depending on the Coupling Facility Resource Management (CFRM) policy.

In the message text:

ssname
is the name of the BatchPipes subsystem.

pipeplex-name
is the name of the BatchPipes pipeplex.

Source: BATCHPIPES

Detecting Module:
ASFPMXSR

System Action: Any job which is using a pipe managed by the BatchPipes subsystems in the pipeplex will be temporarily waited when it attempts to perform processing on the pipe. Once the structure is rebuilt, any jobs that were temporarily waited will be posted and allowed to continue.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP427I BATCHPIPES *ssname* **CROSS-SYSTEM PIPING RESUMED FOR PIPEPLEX** *pipeplex-name***. STRUCTURE REBUILD HAS BEEN STOPPED BECAUSE** *stop-reason-text*

Explanation: Structure rebuild processing to rebuild the coupling facility structure used by the BatchPipes pipeplex, *plex-name*, has been stopped and normal cross-system pipe processing has been resumed for the pipeplex. The reason rebuild processing was stopped is indicated by the *stop-reason-text*.

In the message text:

ssname
is the name of the BatchPipes subsystem.

pipeplex-name
is the name of the BatchPipes pipeplex.

stop-reason-text
One of the following:

A REBUILD STOP WAS REQUESTED BY THE OPERATOR.
A "SETXCF STOP,REBUILD, ..." operator command was issued to stop rebuild processing.

A STRUCTURE FAILURE OCCURRED FOR THE NEW STRUCTURE.
A request to stop rebuild processing was initiated internally by BatchPipes because a structure failure was detected when accessing the new structure.

ONE OR MORE MEMBERS LOST CONNECTIVITY TO THE NEW STRUCTURE.
A request to stop rebuild processing was initiated internally by BatchPipes because at least one or more members of the pipeplex lost connectivity to the new structure. Message ASFP424I is issued on each system that lost connectivity.

ONE OR MORE MEMBERS UNABLE TO CONNECT TO THE NEW STRUCTURE.
A request to stop rebuild processing was initiated internally by BatchPipes because at least one or more members of the pipeplex was unable to connect to the new structure. Message ASFP423I is issued on each system that is unable to connect.

THE SIZE OF THE NEW STRUCTURE IS TOO SMALL.
A request to stop rebuild processing was initiated internally by BatchPipes because the size of the new structure is too small to contain the contents of the old structure.

OF AN UNKNOWN REASON.
The reason the Rebuild Stop request was issued is unknown to BatchPipes stop and rebuild processing.

Source: BATCHPIPES

Detecting Module:
ASFPMXSR

System Action: Any jobs that attempted to perform processing against a pipe during the structure rebuild and were temporarily waited, are now posted and allowed to continue.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP428I BATCHPIPES *ssname* **CROSS-SYSTEM PIPING RESUMED FOR PIPEPLEX** *pipeplex-name***. STRUCTURE REBUILD COMPLETED OK.**

Explanation: Structure rebuild processing to rebuild the coupling facility structure used by the BatchPipes pipeplex, *plex-name*, has been completed successfully and normal cross-system pipe processing has been resumed for the pipeplex.

In the message text:

ssname
is the name of the BatchPipes subsystem.

pipeplex-name
is the name of the BatchPipes pipeplex.

Source: BATCHPIPES

Detecting Module:
ASFPMXSR

System Action: Any jobs that attempted to perform processing against a pipe during the structure rebuild and were temporarily waited, are now posted and allowed to continue.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP429E BATCHPIPES *ssname* CROSS-SYSTEM PIPES HALTED ON *sysname*. *reason-text* THE BATCHPIPES SUBSYSTEM IS BEING CANCELED.

Explanation: Because of the critical error condition indicated by the *reason-text*, the BatchPipes subsystem *ssname* on system *sysname* can no longer continue to service the cross-system pipes that it manages. All cross-system pipe processing has thus been halted.

In the message text:

ssname
is the name of the BatchPipes subsystem.

sysname
is the name of the system on which the BatchPipes subsystem is executing.

reason-text
One of the following:

A BATCHPIPES CF STRUCTURE FAILURE OCCURRED.
either the BatchPipes Coupling Facility Structure or the Coupling Facility on which it resides encountered a failure and is no longer accessible.

LOST CONNECTIVITY TO THE BATCHPIPES CF STRUCTURE.
connectivity to the BatchPipes Coupling Facility structure has been lost. As a result, the structure is no longer accessible.

Source: BATCHPIPES

Detecting Module:
ASFPMXSR

System Action: The BatchPipes subsystem is canceled.

Operator Response: When the coupling facility problem that caused the failure has been fixed, restart the BatchPipes subsystem again.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP430I BATCHPIPES *ssname* ERROR READING PARMLIB MEMBER *asfpbpxx* DUE TO *reason-text*

Explanation: Parmlib member *mname* could not be read from PARMLIB due to the reason indicated by the *reason-text*.

In the message text:

ssname
is the name of the BatchPipes subsystem.

asfpbpxx
is the name of the PARMLIB member that was being read.

reason-text
One of the following:

I/O ERROR
an unrecoverable I/O error occurred when reading the PARMLIB member.

OPEN FAILURE
the BatchPipes subsystem could not open the PARMLIB member.

UNKNOWN ERROR
an unexpected processing error occurred when reading the PARMLIB member.

Source: BATCHPIPES

Detecting Module:
ASFPMIPR

System Action: BatchPipes subsystem processing continues.

If the error occurred during BatchPipes initialization, the default values for the BatchPipes PARMLIB parameters are used. If the error occurred during SET command processing, the parameter values that were previously in affect remain unchanged.

Operator Response: Notify your system programmer.

User Response: None.

Application Programmer Response: None.

System Programmer Response: If the PARMLIB member could not be read because of an UNKNOWN ERROR, contact the IBM Support Center.

If the member could not be read because of an I/O ERROR or OPEN ERROR, ensure that the PARMLIB data set is accessible by the operating system. Then, have the operator use the BatchPipes SET command to specify the PARMLIB member that the BatchPipes subsystem should use for its processing.

ASFP431I BATCHPIPES *ssname* PARMLIB MEMBER *asfpbpxx* NOT FOUND.

Explanation: Parmlib member *mname* could not be found in PARMLIB.

In the message text:

ssname
is the name of the BatchPipes subsystem.

asfpbpxx
is the name of the PARMLIB member that was specified, but could not be found.

Source: BATCHPIPES

Detecting Module:
ASFPMIPR

System Action: BatchPipes subsystem processing continues.

If the error occurred during BatchPipes initialization, the default values for the BatchPipes PARMLIB parameters are used. If the error occurred during SET command processing, the parameter values that were previously in affect remain unchanged.

Operator Response: Determine the correct name of the BatchPipes PARMLIB member and then use the BatchPipes SET command to force the BatchPipes subsystem to use the parameter values specified in that PARMLIB member.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP432I BATCHPIPES *ssname* **PARMLIB MEMBER** *asfpbpxx* **READ.**

Explanation: PARMLIB member *mname* was successfully read and processed.

In the message text:

ssname
is the name of the BatchPipes subsystem.

asfpbpxx
is the name of the PARMLIB member that was read.

Source: BATCHPIPES

Detecting Module:
ASFPMIPR

System Action: BatchPipes subsystem processing continues.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP433I BATCHPIPES *ssname* **PARMLIB MEMBER** *asfpbpxx* **IN ERROR.**
RECORD *nnnn* **SYNTAX IS NOT VALID:**
record-text

Explanation: PARMLIB member *mname* contains a syntax error in record number *nnnn*.

In the message text:

ssname
is the name of the BatchPipes subsystem.

asfpbpxx
is the name of the PARMLIB member that was being read.

nnnn
is the number of the record within the PARMLIB member that is in error.

record-text
is the first 69 characters of the record that is in error.

Source: BATCHPIPES

Detecting Module:
ASFPMC96

System Action: BatchPipes subsystem processing continues.

If the error occurred during BatchPipes initialization, the default values for the BatchPipes PARMLIB parameters are used. If the error occurred during SET command processing, the parameter values that were previously in affect remain unchanged.

Operator Response: Notify your system programmer.

User Response: None.

Application Programmer Response: None.

System Programmer Response: Correct the syntax error. Then, have the operator use the BatchPipes SET command to force the BatchPipes subsystem to use the corrected PARMLIB member for its processing.

ASFP434I BATCHPIPES *ssname* **PARMLIB MEMBER** *asfpbpxx* **IN ERROR.**
VALUE SPECIFIED FOR *pname* **PARAMETER IS OUT OF RANGE:**
record-text

Explanation: The value specified for the *pname* parameter in PARMLIB member *mname* is outside the range of values supported.

In the message text:

ssname
is the name of the BatchPipes subsystem.

asfpbpxx
is the name of the PARMLIB member that was being read.

pname
One of the following:

IDLE
indicates the read/write IDLE inactivity threshold is in error. The valid values supported are 0 to 1440 or OFF.

WAIT
indicates the read/write WAIT inactivity threshold is in error. The valid values supported are 0 to 1440 or OFF.

WAITOPEN
indicates the read/write WAITOPEN inactivity threshold is in error. The valid values supported are 0 to 1440 or OFF.

WAITCLOSE
indicates the read/write WAITCLOSE inactivity threshold is in error. The valid values supported are 0 to 1440 or OFF.

WAITEOF
indicates the read WAITEOF inactivity threshold is in error. The valid values supported are 0 to 1440 or OFF.

MAXBUFNO
indicates the value specified for the subsystem parameter MAXBUFNO is out of range. Allowable values are from 1 to 255.

DEFBUFNO
indicates the value specified for the subsystem parameter DEFBUFNO is out of range. Allowable values are from 1 to MAXBUFNO.

WAITALLOC
indicates the WAITALLOC inactivity threshold is in error. The valid values supported are 0 to 1440 or OFF.

WAITTERM
indicates the WAITTERM inactivity threshold is in error. The valid values supported are 0 to 1440 or OFF.

DB2SSID
indicates the value specified for the subsystem parameter DB2SSID is too long. The subsystem id may be from 1 to 4 characters.

DB2PLAN
indicates the value specified for the subsystem parameter DB2PLAN is too long. The plan name may be from 1 to 8 characters.

record-text
is the first 69 characters of the record that is in error.

Source: BATCHPIPES

Detecting Module:
ASFPMC96

System Action: BatchPipes subsystem processing continues.

If the error occurred during BatchPipes initialization, the default value for the parameter is used. If the error occurred during SET command processing, the value that was previously in affect for the parameter remains unchanged.

Operator Response: Notify your system programmer.

User Response: None.

Application Programmer Response: None.

System Programmer Response: Correct the error. Then have the operator use the BatchPipes SET command to force the BatchPipes subsystem to use the corrected PARMLIB member for its processing.

ASFP435I BATCHPIPES *ssname* PARMLIB MEMBER OPTION *asfpbpxx* IGNORED - SETTINGS OBTAINED FROM PIPEPLEX

Explanation: The parmlib member value is ignored for a Cross-System Subsystem that is joining an existing PipePlex.

In the message text:

ssname
is the name of the BatchPipes subsystem.

asfpbpxx
is the name of the PARMLIB member that was being read.

Source: BATCHPIPES

Detecting Module:
ASFPMIPR

System Action: BatchPipes subsystem processing continues.

The settings are obtained from the active PipePlex.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP436I BATCHPIPES *ssname* **PARMLIB MEMBER** *asfpbpxx* **UPDATE FAILED. UNABLE TO UPDATE** *pname* **PARAMETER:** *record-text*

Explanation: An attempt to use MVS name/token services to update the parameter specified, but either name/token services were not available or the attempt failed.

In the message text:

ssname
is the name of the BatchPipes subsystem.

asfpbpxx
is the name of the PARMLIB member that was being read.

pname
One of the following:

DB2SSID
indicates the DB2SSID parameter was specified.

DB2PLAN
indicates the DB2PLAN parameter was specified.

record-text
is the first 69 characters of the record that is in error.

Source: BATCHPIPES

Detecting Module:
ASFPMC96

System Action: BatchPipes subsystem processing continues.

If the error occurred during BatchPipes initialization, the default value for the parameter is used. If the error occurred during SET command processing, the value that was previously in effect for the parameter remains unchanged.

Operator Response: Notify your system programmer.

User Response: None.

Application Programmer Response: None.

System Programmer Response: If the level of MVS that you are running does not support name/token services, remove the failing key words from the parmlib member. Check the BatchPipes Users Guide and Reference to determine how to update those settings.

ASFP440E BATCHPIPES *ssname* **JOB** *jobname* **WAITING FOR ADDITIONAL ALLOCATIONS ON PIPE** *pname*

Explanation: Job *jobname* using pipe *pname*, has requested allocation. The definition of the pipe connection in the job has specified the ALLOCSYNC option and this connection is now waiting on the allocation of the other job, or jobs, in the pipeline before it will complete allocation. This connection has been in this condition for a period that exceeds the BatchPipes WAITALLOC inactivity threshold.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the job that is waiting.

pname
is the name of the pipe data set on which the job was waiting for additional connections to be allocated.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: The job is allowed to remain on the system.

Operator Response: Notify the appropriate system programmers.

User Response: None.

Application Programmer Response: You can use the BatchPipes ISPF monitor to determine what jobs are active and their current status. Determine if some of the jobs are delayed in the system and ensure that they are progressing as expected.

System Programmer Response: This message may be occurring simply because the WAITALLOC inactivity threshold was set too low. If so, use the BatchPipes SET command to specify the new WAITALLOC inactivity threshold. This message may also occur because WAITALLOC was specified on the SUBSYS statement of the job(s) running under the BatchPipes subsystem. If so, this can only be changed in the JCL on the next invocation of the job(s). Otherwise, issue the BatchPipes STATUS PIPE=*pname* command to see all of the jobs currently processing the pipe. Determine if a problem exists with the suite of jobs, and correct the problems so that the pipe will be closed.

ASFP441I BATCHPIPES *ssname* **JOB** *jobname* **NO LONGER WAITING FOR ADDITIONAL ALLOCATIONS ON PIPE** *pname*

Explanation: The job, *jobname*, has completed allocating the pipe data set, *pname*. Before this message, message ASFP440E was issued.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the pipe job that was waiting.

pname
is the name of the pipe data set on which the job was waiting for additional allocations.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: Action message ASFP440E, which was previously issued to report that the job was held up waiting for additional allocations, is deleted.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP442E BATCHPIPES *ssname* **JOB** *jobname* **WAITING FOR SYNCHRONOUS TERMINATION ON PIPE** *pname*

Explanation: Job *jobname* using pipe *pname*, has completed processing and is in termination. The definition of the pipe connection in the job has specified the TERMSYNC option and this connection is now waiting on the termination of the other job, or jobs, in the pipeline before it will complete termination. This connection has been in this condition for a period that exceeds the BatchPipes WAITTERM inactivity threshold.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the job that is waiting.

pname
is the name of the pipe data set on which the job was waiting for additional connections to terminate.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: The job is allowed to remain on the system.

Operator Response: Notify the appropriate system programmers.

User Response: None.

Application Programmer Response: You can use the BatchPipes ISPF monitor to determine what jobs are active and their current status. Determine if some of the jobs are delayed in the system and ensure that they are progressing as expected.

System Programmer Response: This message may be occurring simply because the WAITTERM inactivity threshold was set too low. If so, use the BatchPipes SET command to specify the new WAITTERM inactivity threshold. This message may also occur because WAITTERM was specified on the SUBSYS statement of the job(s) running under the BatchPipes subsystem. If so, this can only be changed in the JCL on the next invocation of the job(s). Otherwise, issue the BatchPipes STATUS PIPE=*pname* command to see all of the jobs currently processing the pipe. Determine if a problem exists with the suite of jobs, and correct the problems so that the pipe will be closed.

ASFP443I BATCHPIPES *ssname* **JOB** *jobname* **NO LONGER WAITING FOR SYNCHRONOUS TERMINATIONS ON PIPE** *pname*

Explanation: The job, *jobname*, has completed processing. Before this message, message ASFP442E was issued.

In the message text:

ssname
is the name of the BatchPipes subsystem that is managing the pipe data set.

jobname
is the name of the pipe job that was waiting.

pname
is the name of the pipe data set on which the job was waiting for additional jobs to terminate.

Source: BATCHPIPES

Detecting Module:
ASFPMUM

System Action: Action message ASFP442E, which was previously issued to report that the job was held up waiting for additional terminations, is deleted.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP500I TYPE IN THE SUBSYSTEM NAME, WHICH CAN BE UP TO FOUR CHARACTERS IN LENGTH.

Explanation: On panel ASFPPEMN, you must enter the BatchPipes subsystem name for which you want information The subsystem name is up to four characters in length.

Source: BATCHPIPES

Detecting Module:
ASFPPEMN

System Action: The system prompts you for the subsystem name.

Operator Response: None.

User Response: Do one of the following:

- Enter the four-character subsystem name.
- Press **F3** to exit the BatchPipes Monitor.

Application Programmer Response: None.

System Programmer Response: None.

ASFP501I SELECT THE ITEM FOR WHICH YOU WANT STATUS.

Explanation: On panel ASFPPEMN, you must type in a valid number to select the item for which you want status.

Source: BATCHPIPES

Detecting Module:
ASFPPEMN

System Action: The system prompts you for the item number.

Operator Response: None.

User Response: Do one of the following:

- Enter the one-digit item number and press Enter. The item numbers are:
  1. Subsystem
  2. Thresholds
  3. Job(s)
  4. Pipe(s)
  5. Topology
- Press **F3** to exit the BatchPipes Monitor.

Application Programmer Response: None.

System Programmer Response: None.

ASFP502I ENTER ONE OF THE FOLLOWING: THE JOB NAME, ? FOR A SELECTION LIST, ABC* FOR ALL JOBS PREFIXED WITH ABC, OR * FOR ALL JOBS.

Explanation: On panel ASFPPEJI, you must type in one or more job names for which you want job information.

Source: BATCHPIPES

Detecting Module:
ASFPMEJB

System Action: The system prompts you for a job name.

Operator Response: None.

User Response: Do one of the following:

- Enter the job name in one of the following formats and press Enter:
  - The eight-character job name or names
  - **?** to obtain a selection list of job names
  - **ABC*** for all jobs prefixed with ABC
  - ***** for all jobs
- Press **F3** to exit the selection panel.

Application Programmer Response: None.

System Programmer Response: None.

ASFP503I ISPF ERROR: ISPF COULD NOT *action* PANEL *panelid*

Explanation: ISPF experienced a system error when trying to perform an action on a panel.

In the message text:

action
One of the following:

ADD A POP-UP FOR
ISPF could not add a pop-up for the panel.

DISPLAY
ISPF could not display the panel.

REMOVE THE POP-UP FOR
ISPF could not remove the pop-up for the panel.

panelid
The panelid of the panel for which the action could not be performed.

Source: BATCHPIPES

Detecting Module:

ASFPMEHO	ASFPMEJB	ASFPMEMN	ASFPMEPI
ASFPMESU	ASFPMETO		

System Action: Processing continues.

Operator Response: None.

User Response: Keep pressing **F3** until you exit from the BatchPipes Monitor and are back in ISPF. Then type **TSO ASFPMEMN** from ISPF to start the BatchPipes Monitor again. If the same message appears again, contact IBM service personnel to report the text of the message.

Application Programmer Response: None.

System Programmer Response: None.

ASFP504I ENTER ONE OF THE FOLLOWING: THE PIPE NAME, ? FOR A SELECTION LIST, ABCD* FOR ALL PIPES PREFIXED WITH ABCD, OR * FOR ALL PIPES.

Explanation: On panel ASFPPEPI, you must type in one or more pipe names for which you want pipe information.

Source: BATCHPIPES

Detecting Module:
ASFPMEPI

System Action: The system prompts you for a pipe name.

Operator Response: None.

User Response: Do one of the following:

- Enter the pipe name in one of the following formats and press Enter:
  - The pipe name or names which can be up to 44 characters in length
  - **?** to obtain a selection list of pipe names
  - **ABCD*** for all jobs prefixed with ABCD
  - ***** for all pipes
- Press **F3** to exit the selection panel.

Application Programmer Response: None.

System Programmer Response: None.

ASFP505I *type* **LIST COULD NOT BE OBTAINED.** *reason*

Explanation: The BatchPipes Monitor was unable to retrieve a selection list of job or pipe names.

In the message text:

type
The type of list, which is either:

JOB
PIPE

reason
One of the following:

THE SUBSYSTEM IS ACTIVE, BUT THERE ARE NO {PIPES|JOBS} ASSOCIATED WITH THE SUBSYSTEM.

THE SUBSYSTEM IS NOT ACTIVE.

THERE WAS A SYSTEM ERROR IN RETRIEVING THE DATA.

Source: BATCHPIPES

Detecting Module:
ASFPMEJB ASFPMEPI

System Action: Processing continues.

Operator Response: None.

User Response: If *reason* is **there was a system error in retrieving the data**, contact IBM service personnel.

Application Programmer Response: None.

System Programmer Response: None.

ASFP506I NO JOB INFORMATION IS AVAILABLE *[reason]*

Explanation: For the job name or names typed or selected, no job information is available.

In the message text:

[reason]
One of the following:

BECAUSE THERE ARE NO JOBS ASSOCIATED WITH THE SUBSYSTEM
There are no jobs running in the specified subsystem.

BECAUSE THE SUBSYSTEM IS NOT ACTIVE
The subsystem is not active.

BECAUSE THERE WAS A SYSTEM ERROR IN RETRIEVING THE DATA
There was a system error in retrieving the job information.

Source: BATCHPIPES

Detecting Module:
ASFPMEPI

System Action: Processing continues.

Operator Response: None.

User Response: If *reason* is **there was a system error in retrieving the data**, contact IBM service personnel.

Application Programmer Response: None.

System Programmer Response: None.

ASFP507I NO PIPE INFORMATION IS AVAILABLE *[reason]*

Explanation: For the pipe name or names typed or selected, no pipe information is available.

In the message text:

[reason]
One of the following:

BECAUSE THERE ARE NO PIPES ASSOCIATED WITH THE SUBSYSTEM
There are currently no pipes in the specified subsystem.

BECAUSE THE SUBSYSTEM IS NOT ACTIVE
The subsystem is not active so there is no pipe information available.

BECAUSE THERE WAS A SYSTEM ERROR IN RETRIEVING THE DATA
There was a system error in retrieving the pipe information.

Source: BATCHPIPES

Detecting Module:
ASFPMEPI

System Action: Processing continues.

Operator Response: None.

User Response: If *reason* is **there was a system error in retrieving the data**, contact IBM service personnel.

Application Programmer Response: None.

System Programmer Response: None.

ASFP508I NO *ssname* **SUBSYSTEM INFORMATION IS AVAILABLE** *[reason]*

Explanation: For the subsystem name typed, no information is available.

In the message text:

ssname
The name of the subsystem for which there was no information available.

[reason]
One of the following:

BECAUSE THE SUBSYSTEM IS NOT ACTIVE
The subsystem is not active so there is no information available.

BECAUSE THRESHOLDS ARE NOT ACTIVE
Threshold monitoring is not active so there is no information available.

BECAUSE THERE WAS A SYSTEM ERROR IN RETRIEVING THE DATA
There was a system error in retrieving the subsystem information.

Source: BATCHPIPES

Detecting Module:
ASFPMESU

System Action: Processing continues.

Operator Response: None.

User Response: If *reason* is **there was a system error in retrieving the data**, contact IBM service personnel.

Application Programmer Response: None.

System Programmer Response: None.

ASFP509I *ssname* **IS NOT A VALID SUBSYSTEM NAME: ENTER A VALID NAME.**

Explanation: The subsystem name typed is not a valid subsystem name.

In the message text:

ssname
The subsystem name that was specified as input.

Source: BATCHPIPES

Detecting Module:
ASFPMEMN

System Action: Processing continues.

Operator Response: None.

User Response: Enter a valid subsystem name. If the name that was entered is a valid name, contact IBM service personnel.

Application Programmer Response: None.

System Programmer Response: None.

ASFP510I NO TOPOLOGY INFORMATION IS AVAILABLE FOR *type name [reason]*

Explanation: For the job or pipe name typed or selected, no topology information is available.

In the message text:

type
Either:

JOB
PIPE

name
The name of the job or pipe, depending on the *type*, for which there was no topology information available.

[reason]
One of the following:

BECAUSE THE SUBSYSTEM IS NOT ACTIVE
The subsystem is not active so there is no topology information available.

BECAUSE THERE WAS A SYSTEM ERROR IN RETRIEVING THE DATA
There was a system error in retrieving the topology information.

Source: BATCHPIPES

Detecting Module:
ASFPMETO

System Action: Processing continues.

Operator Response: None.

User Response: If *reason* is **there was a system error in retrieving the data**, contact IBM service personnel.

Application Programmer Response: None.

System Programmer Response: None.

ASFP511I NO THRESHOLD INFORMATION IS AVAILABLE FOR SUBSYSTEM *ssname [reason]*

Explanation: For the subsystem name typed, no threshold information is available.

In the message text:

ssname
The subsystem for which there was no threshold information available.

[reason]
One of the following:

BECAUSE THE SUBSYSTEM IS NOT ACTIVE
The subsystem is not active so there is no threshold information available.

BECAUSE THERE WAS A SYSTEM ERROR IN RETRIEVING THE DATA
There was a system error in retrieving the threshold information.

Source: BATCHPIPES

Detecting Module:
ASFPMEHO

System Action: Processing continues.

Operator Response: None.

User Response: If *reason* is **there was a system error in retrieving the data**, contact IBM service personnel.

Application Programmer Response: None.

System Programmer Response: None.

ASFP512I NOTHING WAS SELECTED ON PANEL *panelid*

Explanation: There was nothing selected on the selection panel displayed.

In the message text:

panelid
The panelid of the selection panel from which nothing was selected.

Source: BATCHPIPES

Detecting Module:
ASFPMEJB ASFPMEPI ASFPMETO

System Action: Processing continues.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP513I ISPF ERROR: ISPF COULD NOT *action* **TABLE** *tablnam*

Explanation: ISPF could not perform an action on an ISPF table due to a system error.

In the message text:

action
One of the following:

ADD A ROW TO
ISPF could not add a row to the table.

CREATE
ISPF could not create the table.

DISPLAY
ISPF could not display the table.

GO TO THE TOP OF
ISPF could not go to the top of the table.

MODIFY A ROW OF
ISPF could not modify a row of the table.

tablnam
The name of the table for which the action could not be performed.

Source: BATCHPIPES

Detecting Module:
ASFPMEHO ASFPMEJB ASFPMEMN ASFPMEPI
ASFPMESU ASFPMETO

System Action: Processing continues.

Operator Response: None.

User Response: Keep pressing **F3** until you exit from the BatchPipes Monitor and are back in ISPF. Then type **TSO ASFPMEMN** from ISPF to start the BatchPipes Monitor again. If the same message appears again, contact IBM service personnel to report the text of the message.

Application Programmer Response: None.

System Programmer Response: None.

ASFP514I SELECT A SORT FIELD.

Explanation: The user pressed Enter on selection panel ASFPPEFS without selecting a sort field.

Source: BATCHPIPES

Detecting Module:
ASFPMEJB ASFPMEPI

System Action: Processing continues.

Operator Response: None.

User Response: Do one of the following:

- Select a sort field by typing **S** next to the field.
- Press **F3** or **F12** to exit from the selection panel.

Application Programmer Response: None.

System Programmer Response: None.

ASFP515I SELECT ONLY ONE SORT FIELD.

Explanation: The user selected more than one sort field on selection panel ASFPPEFS.

Source: BATCHPIPES

Detecting Module:
ASFPMEJB ASFPMEPI

System Action: Processing continues.

Operator Response: None.

User Response: Do one of the following:

- Select a single sort field by typing **S** next to the field.
- Press **F3** or **F12** to exit from the selection panel.

Application Programmer Response: None.

System Programmer Response: None.

ASFP516I CONCATENATION TO LIBRARY *library* FAILED.

Explanation: The system could not concatenate to a library needed to run the BatchPipes Monitor.

In the message text:

library
The library for which the concatenation failed.

Source: BATCHPIPES

Detecting Module:
ASFPMEMN

System Action: Processing ends.

Operator Response: None.

User Response: Contact your system programmer.

Application Programmer Response: None.

System Programmer Response: Ensure all libraries required to run the BatchPipes Monitor are available to the system.

ASFP517I *ssname* IS A VALID SUBSYSTEM, BUT IT IS NOT ACTIVE.

Explanation: The subsystem name typed is a valid subsystem, but this subsystem is not active.

In the message text:

ssname
The subsystem name that was typed as input.

Source: BATCHPIPES

Detecting Module:
ASFPMEMN

System Action: Processing continues.

Operator Response: None.

User Response: If the subsystem should be active, notify operations personnel in your installation.

Application Programmer Response: None.

System Programmer Response: None.

ASFP518I ONE OR MORE *type* NAMES ARE NOT VALID. EACH INCORRECT NAME IS MARKED WITH ?S.

Explanation: One or more job or pipe names typed are not valid.

In the message text:

type
Either:

JOB
PIPE

Source: BATCHPIPES

Detecting Module:
ASFPMEJB ASFPMEPI ASFPMETO

System Action: The system marks each incorrect job or pipe name or names with **?**s on the panel on which they were typed.

Operator Response: None.

User Response: Do one of the following:

- Erase the incorrect job or pipe names.
- Type a single **?** in either the job or pipe name field to get a selection list of valid job or pipe names.
- Type correct job or pipe names over the incorrect names.

Application Programmer Response: None.

System Programmer Response: None.

ASFP519I SUBSYSTEM *ssname* IS *state*

Explanation: The subsystem is either stopping or not active. If the subsystem is stopping, the requested data will still be displayed.

In the message text:

ssname
The name of the subsystem.

state
Either:

STOPPING
NOT ACTIVE

Source: BATCHPIPES

Detecting Module:
ASFPMEMN

System Action: Processing continues.

Operator Response: None.

User Response: None

Application Programmer Response: None.

System Programmer Response: None.

ASFP520I SELECT A JOB OR TYPE A FOR ALL JOBS.

Explanation: The user pressed Enter while on selection panel ASFPPEJS without selecting a job name.

Source: BATCHPIPES

Detecting Module:
ASFPMEJB

System Action: Processing continues.

Operator Response: None.

User Response: Do one of the following:

- Type **A** in the appropriate field to select all jobs.
- Select one or more job names by typing **S** next to the job name or names.
- Press **F3** or **F12** to exit from the selection panel.

Application Programmer Response: None.

System Programmer Response: None.

ASFP521I SELECT A PIPE OR TYPE A FOR ALL JOBS.

Explanation: The user pressed Enter while on selection panel ASFPPEPS without selecting a pipe name.

Source: BATCHPIPES

Detecting Module:
ASFPMEPI

System Action: Processing continues.

Operator Response: None.

User Response: Do one of the following:

- Type **A** in the appropriate field to select all pipes.
- Select one or more pipe names by typing **S** next to the pipe name or names.
- Press **F3** or **F12** to exit from the selection panel.

Application Programmer Response: None.

System Programmer Response: None.

ASFP522I SELECT ONLY ONE JOB NAME.

Explanation: The user selected more than one job name on selection panel ASFPPEJX.

Source: BATCHPIPES

Detecting Module:
ASFPMETO

System Action: Processing continues.

Operator Response: None.

User Response: Do one of the following:

- Select a single job name by typing **S** next to the job name.
- Press **F3** or **F12** to exit from the selection panel.

Application Programmer Response: None.

System Programmer Response: None.

ASFP523I SELECT ONLY ONE PIPE NAME.

Explanation: The user selected more than one pipe name on selection panel ASFPPEPX.

Source: BATCHPIPES

Detecting Module:
ASFPMETO

System Action: Processing continues.

Operator Response: None.

User Response: Do one of the following:

- Select a single pipe name by typing **S** next to the pipe name.
- Press **F3** or **F12** to exit from the selection panel.

Application Programmer Response: None.

System Programmer Response: None.

ASFP524I SELECT A PIPE NAME.

Explanation: The user pressed Enter on selection panel ASFPPEPX without selecting a pipe name.

Source: BATCHPIPES

Detecting Module:
ASFPMETO

System Action: Processing continues.

Operator Response: None.

User Response: Do one of the following:

- Select a pipe name by typing **S** next to the name.
- Press **F3** or **F12** to exit from the selection panel.

Application Programmer Response: None.

System Programmer Response: None.

ASFP525I SELECT A JOB NAME.

Explanation: The user pressed Enter on selection panel ASFPPEJX without selecting a job name.

Source: BATCHPIPES

Detecting Module:
ASFPMETO

System Action: Processing continues.

Operator Response: None.

User Response: Do one of the following:

- Select a job name by typing **S** next to the name.
- Press **F3** or **F12** to exit from the selection panel.

Application Programmer Response: None.

System Programmer Response: None.

ASFP526I TYPE A PIPE OR A JOB NAME OR ENTER A ? IN EITHER FIELD TO OBTAIN A SELECTION LIST OF PIPE OR JOB NAMES.

Explanation: The user pressed Enter on panel ASFPPEPJ without typing a pipe or job name.

Source: BATCHPIPES

Detecting Module:
ASFPMETO

System Action: Processing continues.

Operator Response: None.

User Response: Do one of the following:

- Type a pipe name, which can be up to 44 characters in length.
- Type **?** in the pipe name field to obtain a selection list of pipe names. list of pipe names.
- Type a job name, which is eight characters in length.
- Type **?** in the job name field to obtain a selection list of job names.
- Press **F3** or **F12** to exit from the panel.

Application Programmer Response: None.

System Programmer Response: None.

ASFP527I THE VALUE TYPED IN THE SORT FIELD IS INCORRECT. TYPE A CORRECT VALUE OR TYPE ? TO OBTAIN A SELECTION LIST OF VALID SORT FIELDS..

Explanation: The user typed an incorrect value in the **sort field**.

Source: BATCHPIPES

Detecting Module:
ASFPMEJB ASFPMEPI

System Action: Processing continues.

Operator Response: None.

User Response: Do one of the following:

- In the **Sort field**, type either:
  - A valid sort field value
  - **?** to obtain a selection list of valid sort fields
- Erase the data in the sort field, if you do not wish to sort the data.

Application Programmer Response: None.

System Programmer Response: None.

ASFP528I SORT ORDER IS EITHER A OR D: ASCENDING OR DESCENDING.

Explanation: The user typed an incorrect value in the **A** or **D** field.

Source: BATCHPIPES

Detecting Module:
ASFPMEJB ASFPMEPI

System Action: Processing continues.

Operator Response: None.

User Response: In the **A or D** field, type either:

- **A** (for ascending)
- **D** (for descending)

Application Programmer Response: None.

System Programmer Response: None.

ASFP529I SUPPLY A SORT FIELD IN ADDITION TO A SORT ORDER.

Explanation: The user supplied a sort order (**A** or **D**), but did not supply a sort field. Both are required to sort the table.

Source: BATCHPIPES

Detecting Module:
ASFPMEJB ASFPMEPI

System Action: Processing continues.

Operator Response: None.

User Response: Do one of the following:

- In the **Sort field**, type either:
  - A valid sort field value
  - **?** to obtain a selection list of valid sort fields
- Erase the data in the sort direction field, if you do not wish to sort the data.

Application Programmer Response: None.

System Programmer Response: None.

ASFP960I BATCHPIPES *ssname* SUBSYSTEM FUNCTIONS DISABLED.

Explanation: In the process of ending either normally or abnormally, the BatchPipes subsystem *ssname* disabled the subsystem functions it normally provides for jobs that use pipes managed by the subsystem.

In the message text:

ssname
is the name of the BatchPipes subsystem.

Source: BATCHPIPES

Detecting Module:
ASFPMSSD

System Action: The BatchPipes subsystem will end. Any jobs that attempt to allocate, open, or close pipes managed by the subsystem *ssname* will fail allocation, open, or close processing.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP962I BATCHPIPES *ssname* SUBSYSTEM FUNCTION DISABLEMENT FAILED.

Explanation: In the process of ending either normally or abnormally, the BatchPipes subsystem *ssname* attempted to disable the subsystem functions it normally provides for jobs that use pipes managed by the subsystem. However, a failure occurred that prevented the subsystem from completely disabling all the subsystem functions it supports.

In the message text:

ssname
is the name of the BatchPipes subsystem.

Source: BATCHPIPES

Detecting Module:
ASFPMSSD

System Action: The BatchPipes subsystem will end. Any jobs that attempt to allocate, open, or close pipes managed by the subsystem *ssname* may abnormally end in error.

Operator Response: None.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

ASFP963I BATCHPIPES *ssname1* CLEANUP OF THE CONNECTION TO PIPEPLEX *pipeplex-name* FOR BATCHPIPES SUBSYSTEM *ssname2* ON SYSTEM *sysname* FAILED. THE CONNECTION IS CONSIDERED UNRECOVERABLE.

Explanation: Following the ending of the Batchpipes subsystem, *ssname2*, on system, *sysname*, the BatchPipes subsystem, *ssname1*, on the system that issued this message attempted to cleanup the ended subsystem's connection to the pipeplex, but failed. After several unsuccessful cleanup attempts, the connection was finally marked as unrecoverable.

In the message text:

ssname1
is the name of the BatchPipes subsystem that was performing the cleanup and issued this message.

pipeplex-name
is the name of a BatchPipes pipeplex.

ssname2
is the name of the ended BatchPipes subsystem whose connection to the pipeplex cannot be cleaned up and thus it has been set unrecoverable.

sysname
is the name of the system on which the ended BatchPipes subsystem. was running.

Source: BATCHPIPES

Detecting Module:
ASFPMXTT

System Action: Because the pipeplex connection for the ended BatchPipes subsystem could not be successfully cleaned up, data related to that instance of the subsystem on the indicated system might remain in existence in the BatchPipes coupling facility structure and may potentially cause problems if a new instance of the same subsystem on the same system joins the pipeplex. As a result, any new instance of the BatchPipes subsystem, *ssname2*, on system, *sysname*, will not be allowed to join the indicated BatchPipes pipeplex.

Operator Response: To cleanup the pipeplex connection for the ended subsystem so a new instance of the subsystem can be started on the same system, the entire pipeplex will have to be quiesced. To quiesce the pipeplex, each BatchPipes subsystem that is a member of the indicated pipeplex must be either stopped or canceled. This should be done at a point in time that will have the least impact on your installation's production work.

User Response: None.

Application Programmer Response: None.

System Programmer Response: None.

BatchPipes Message Routing & Descriptor Codes

The following is a list of the routing and descriptor codes for all the BatchPipes messages. Messages that are returned back to callers of the subsystem interface (such as, allocation) and messages that are issued by the BatchPipes monitor (ASFP5xxI messages) do not have any routing or descriptor code associated with them.

Message Identifier	Routing Code	Descriptor Code
ASFP000I	2	4
ASFP001E	2	3
ASFP005I	2,10	12
ASFP006I	2	12
ASFP007I	2	4
ASFP008I	2,10	12
ASFP009E	2	4
ASFP011I	2	4
ASFP012I	2	4
ASFP013I	2,10	12
ASFP014I	2	12
ASFP016I	2,10	12
ASFP017I	2	12
ASFP018I	2,10	4
ASFP019I	2,10	4
ASFP020I	2	12
ASFP021I	2,10	12
ASFP022I	2,10	12
ASFP023I	2,10	12
ASFP024I	2	4
ASFP033E	2	12
ASFP081A	2	2
ASFP082I	2	12
ASFP083I	2	12
ASFP084I	2	12
ASFP201I	-	5
ASFP202I	-	5
ASFP203I	-	5
ASFP204I	-	5
ASFP206I	-	5
ASFP207I	-	5
ASFP209I	-	5
ASFP210I	-	5
ASFP211I	-	5
ASFP212I	-	5
ASFP213I	-	5
ASFP216I	-	5
ASFP217I	-	5
ASFP218I	-	5
ASFP220I	-	5
ASFP221I	-	5
ASFP230I	-	5
ASFP231I	-	5
ASFP240I	-	5
ASFP241I	-	5
ASFP242I	-	5
ASFP243I	-	5
ASFP310I	-	-
ASFP311I	-	-
ASFP312I	11	-
ASFP314I	-	-
ASFP315I	-	-
ASFP318I	-	-
ASFP320I	11	-
ASFP321I	11	-
ASFP322I	11	-
ASFP324I	11	-
ASFP325I	11	-
ASFP326I	-	-
ASFP327I	11	-
ASFP328I	11	-
ASFP329I	11	-
ASFP330I	-	-
ASFP331I	11	-
ASFP332I	11	-
ASFP333I	11	-
ASFP334I	10,11	6
ASFP335I	11	-
ASFP336I	10,11	6
ASFP337I	11	-
ASFP338I	11	-
ASFP339I	11	-
ASFP340I	11	-
ASFP341I	11	-
ASFP342I	11	-
ASFP343I	11	-
ASFP344I	11	-
ASFP345I	11	-
ASFP349I	11	-
ASFP350I	11	-
ASFP351I	11	-
ASFP352I	11	-
ASFP353I	11	-
ASFP354I	11	-
ASFP355I	11	-
ASFP356I	11	-
ASFP357I	11	-
ASFP358I	11	-
ASFP359I	11	-
ASFP360I	11	-

Message Identifier	Routing Code	Descriptor Code
ASFP361I	11	-
ASFP362I	11	-
ASFP363I	11	-
ASFP364E	11	-
ASFP365I	11	-
ASFP366E	-	-
ASFP367E	-	-
ASFP368E	-	-
ASFP369E	-	-
ASFP370I	11	-
ASFP371I	11	-
ASFP374I	11	-
ASFP375I	11	-
ASFP376I	11	-
ASFP377I	11	-
ASFP378I	11	-
ASFP390I	11	-
ASFP391I	11	-
ASFP392I	11	-
ASFP393I	11	-
ASFP394I	11	-
ASFP395I	11	-
ASFP396I	11	-
ASFP399I	-	-
ASFP401E	1	11
ASFP402I	2	6
ASFP403E	1	11
ASFP404I	2	6
ASFP405E	1	11
ASFP406I	2	6
ASFP407E	1	11
ASFP408I	2	6
ASFP409E	1	11
ASFP410I	2	6
ASFP411E	1	11
ASFP412I	2	6
ASFP413E	1	11
ASFP414E	1	11
ASFP415E	1	11
ASFP417E	1	11
ASFP418I	2	6
ASFP419I	2	6
ASFP420I	2	6
ASFP422I	2	6
ASFP423I	2,10	12
ASFP424I	2,10	12
ASFP425E	2,10	7,11
ASFP426I	2,10	7,11
ASFP427I	2,10	12
ASFP428I	2,10	12
ASFP429E	2,10	7,11
ASFP430I	2	4,5
ASFP431I	2	4,5
ASFP432I	2	4,5
ASFP433I	2,10	4
ASFP434I	2,10	4
ASFP435I	2	4,5
ASFP436I	2,10	4
ASFP440E	1	11
ASFP441I	2	6
ASFP442E	1	11
ASFP443I	2	6
ASFP500I	-	-
ASFP501I	-	-
ASFP502I	-	-
ASFP503I	-	-
ASFP504I	-	-
ASFP505I	-	-
ASFP506I	-	-
ASFP507I	-	-
ASFP508I	-	-
ASFP509I	-	-
ASFP510I	-	-
ASFP511I	-	-
ASFP512I	-	-
ASFP513I	-	-
ASFP514I	-	-
ASFP515I	-	-
ASFP516I	-	-
ASFP517I	-	-
ASFP518I	-	-
ASFP519I	-	-
ASFP520I	-	-
ASFP521I	-	-
ASFP522I	-	-
ASFP523I	-	-
ASFP524I	-	-
ASFP525I	-	-
ASFP526I	-	-
ASFP527I	-	-
ASFP528I	-	-
ASFP529I	-	-
ASFP960I	2	4
ASFP962I	2,10	4
ASFP963I	2,10	12

Appendix E: BatchPipes Codes

The following BatchPipes reason codes are used with BatchPipes **BC6** ABENDs to indicate the reason for the ABEND.

If you have an ABEND with a reason code not defined in the following set of reason codes, have your system programmer search the problem reporting data bases for a fix for the problem. If no fix exists, contact the IBM Support Center, providing a copy of the SYSLOG, the SVC dump, and SYS1.LOGREC error records.

C9384830

Explanation: BatchPipes Initialization: Start BatchPipes - SSCT missing or invalid (IEFSSNxx problem).

Source: BatchPipes

System Action: The BatchPipes subsytem terminates.

User Response: None.

Operator Response: Verify correct subsystem name, restart BatchPipes. If subsystem name is correct contact your system programmer; save dump.

System Programmer Response: Subsystem name for command should match the SSN used for BatchPipes. Verify that the correct subsystem name is in the IEFSSNxx and in use for the current IPL. If correct SSN, notify IBM BatchPipes service contact.

D7561490

Explanation: BatchPipes Pipe-manager: Maximum number of pipes allocated.

Source: BatchPipes

System Action: The Application is abnormally terminated.

User Response: Resubmit job.

Operator Response: Perform local procedures to restrict initiation of jobs using pipes. Verify number of pipes with BatchPipes STATUS command.

System Programmer Response: None.

D7561890

Explanation: BatchPipes Pipe-manager: Maximum connections allowed exceeded.

Source: BatchPipes

System Action: The Application is abnormally terminated.

User Response: Resubmit job.

Operator Response: Perform local procedures to restrict initiation of jobs using pipes. Verify number of connections with BatchPipes STATUS command.

System Programmer Response: None.

D7569430

Explanation: An asynchronous cross-memory POST failed.

Source: BatchPipes

System Action: The application is abnormally terminated.

User Response: Notify your system programmer. Resubmit jobs.

Operator Response: Restart BatchPipes.

System Programmer Response: If a job waiting on a pipe empty or pipe full condition is cancelled or terminates before a post (indicating relief of the empty of full condition), a cross memory post may fail.

D7642030

Explanation: Initialization Routine - Too many pipes and connections requested (not enough data space blocks)

Source: BatchPipes

System Action: The BatchPipes subsystem is abnormally terminated.

User Response: None.

Operator Response: Attempt BatchPipes restart. If same failure occurs the second time, notify your system programmer.

System Programmer Response: Search the problem reporting data bases for a fix for the problem. If no fix exists, contact the IBM Support Center, providing a copy of the SYSLOG, the SVC dump, and the SYS1.LOGREC error records.

E25E2030

Explanation: Get PC-cp Routine - caller-provided RPL, ACB, DEB chain invalid.

Source: BatchPipes

System Action: The application is abnormally terminated.

User Response: Possible attempted integrity violation. Notify system programmer.

Operator Response: Possible attempted integrity violation. Notify system programmer.

System Programmer Response: Verify application is using standard interfaces and is not attempting to penetrate the BatchPipes subsytem. If otherwise, contact the IBM BatchPipes service support contact.

E2662030

Explanation: BatchPipes Subsystem: Put PC-cp Routine - caller-provided RPL, ACB, DEB chain invalid.

Source: BatchPipes

System Action: The application is abnormally terminated.

User Response: Possible attempted integrity violation. Notify system programmer.

Operator Response: Possible attempted integrity violation. Notify system programmer.

System Programmer Response: Verify application is using standard interfaces and is not attempting to penetrate the BatchPipes subsytem. If otherwise, notify the IBM BatchPipes service support contact.

E2C90130

Explanation: A WRITE or PUT request was incorrectly issued against a pipe that is open for INPUT. The request cannot be processed.

System Action: The application is abnormally terminated.

User Response: Ensure the application issues the correct type of data operation (i.e., GET/READ). Then resubmit the job. If the problem persists, notify your system programmer.

Operator Response: None.

System Programmer Response: Search the problem reporting data bases for a fix for the problem. If no fix exists, contact the IBM Support Center, providing a copy of the SYSLOG, the SVC dump, and the SYS1.LOGREC error records.

E2C90230

Explanation: A READ or GET request was incorrectly issued against a pipe that is open for OUTPUT. The request cannot be processed.

Source: BatchPipes

System Action: The application is abnormally terminated.

User Response: Ensure the application issues the correct type of data operation (i.e., PUT/WRITE). Then resubmit the job. If the problem persists, notify your system programmer.

Operator Response: None.

System Programmer Response: Search the problem reporting data bases for a fix for the problem. If no fix exists, contact the IBM Support Center, providing a copy of the SYSLOG, the SVC dump, and the SYS1.LOGREC error records.

E2CA0130

Explanation: A READ or GET request was incorrectly issued against a pipe that is open for OUTPUT. The request cannot be processed.

Source: BatchPipes

System Action: The application is abnormally terminated.

User Response: Ensure the application issues the correct type of data operation (i.e., PUT/WRITE). Then resubmit the job. If the problem persists, notify your system programmer.

Operator Response: None.

System Programmer Response: Search the problem reporting data bases for a fix for the problem. If no fix exists, contact the IBM Support Center, providing a copy of the SYSLOG, the SVC dump, and the SYS1.LOGREC error records.

E2CB0130

Explanation: A READ or GET request was incorrectly issued against a pipe that is open for OUTPUT. The request cannot be processed.

Source: BatchPipes

System Action: The application is abnormally terminated.

User Response: Ensure the application is issuing the correct type of data operation (i.e., PUT/WRITE). Then resubmit the job. If the problem persists, notify your system programmer.

Operator Response: None.

System Programmer Response: Search the problem reporting data bases for a fix for the problem. If no fix exists, contact the IBM Support Center, providing a copy of the SYSLOG, the SVC dump, and the SYS1.LOGREC error records.

E2CC0130

Explanation: A WRITE or PUT request was incorrectly issued against a pipe that is open for INPUT. The request cannot be processed.

Source: BatchPipes

System Action: The application is abnormally terminated.

User Response: Ensure the application issues the correct type of data operation (i.e., GET/READ). Then resubmit the job. If the problem persists, notify your system programmer.

Operator Response: None.

System Programmer Response: Search the problem reporting data bases for a fix for the problem. If no fix exists, contact the IBM Support Center, providing a copy of the SYSLOG, the SVC dump, and the SYS1.LOGREC error records.

E2CD0130

Explanation: A WRITE or PUT request was incorrectly issued against a pipe that is open for INPUT. The request cannot be processed.

Source: BatchPipes

System Action: The application is abnormally terminated.

User Response: Ensure the application issues the correct type of data operation (i.e., GET/READ). Then resubmit the job. If the problem persists, notify your system programmer.

Operator Response: None.

System Programmer Response: Search the problem reporting data bases for a fix for the problem. If no fix exists, contact the IBM Support Center, providing a copy of the SYSLOG, the SVC dump, and the SYS1.LOGREC error records.

E2D62030

Explanation: Open/Close exit Routine - DEB Access type not supported (not INPUT nor OUTPUT).

Source: BatchPipes

System Action: The application is abnormally terminated.

User Response: Verify the OPEN is not supported by BatchPipes. If it is, notify the IBM BatchPipes service support contact.

Operator Response: None.

System Programmer Response: None.

E2D62040

Explanation: BatchPipes Subsys: Open/ Close exit Routine - ACB OPEN STYLE not supported (not a SAM ACB).

Source: BatchPipes

System Action: The application is abnormally terminated.

User Response: Ensure the application is opening the pipe with one of the supported options of INPUT or OUTPUT and then resubmit the job(s). If the problem persists, notify your system programmer.

Operator Response: None.

System Programmer Response: Search the problem reporting data bases for a fix for the problem. If no fix exists, contact the IBM Support Center, providing a copy of the SYSLOG, the SVC dump, and the SYS1.LOGREC error records.

E2D62050

Explanation: BatchPipes Subsys: Open/ Close exit Routine - Multiple opens attempted for a single ddname.

Source: BatchPipes

System Action: The application is abnormally terminated.

User Response: Ensure the application issues only one open against a single ddname and then resubmit the job(s). If the problem persists, notify your system programmer.

Operator Response: None.

System Programmer Response: Search the problem reporting data bases for a fix for the problem. If no fix exists, contact the IBM Support Center, providing a copy of the SYSLOG, the SVC dump, and the SYS1.LOGREC error records.

Appendix F: Understanding Syntax Diagrams

This appendix describes the syntax representation, known as “railroad tracks,” that is used throughout this book.

Syntax is described using the structure defined below.

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

  The ►►─── symbol indicates the beginning of a stage function or subparameter.

  The ───◄ symbol indicates that the syntax is continued on the next line.

  The ►─── symbol indicates a continuation from the previous line.

  The ───►◄ symbol indicates the end of a stage function or subparameter.

  An *italicized* lower-case word indicates a variable.

- Required items appear on the horizontal line (the main path).

```
►►──KEYword──reqiured_item──►◄
```

  Capitalization of the keywords indicates the shorthand version of the keyword. Using this example, you can use “KEYWORD” or “KEY.”

- Optional items appear below the main path.

```
►►──KEYword──┬───────────────┬──►◄
             └─reqiured_item─┘
```

- If you can choose from two or more items, they appear vertically, in a stack.

  If you must choose one of the items, one item of the stack appears on the main path.

```
►►──KEYword──┬─reqiured_choice1─┬──►◄
             └─reqiured_choice2─┘
```

  If choosing one of the items is optional, the entire stack appears below the main path.

```
►►──KEYword──┬──────────────────┬──►◄
             ├─optional_choice1─┤
             └─optional_choice2─┘
```

  If one of the items has a default, it appears above the main path and the overriding choices will be shown below the line.

```
                 ┌─default──────────┐
►►──KEYword──────┼──────────────────┼──►◄
                 ├─optional_choice1─┤
                 └─optional_choice2─┘
```

- An arrow returning to the left above the main line indicates an item that can be repeated indefinitely.

```
                  ┌───────────────┐
                  ▼               │
►►──KEYword──────repeatable_item──┴──►◄
```

A repeat arrow with a syntax note indicates how many times this can be repeated.

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

Note:
[1] Specify the <parameter> 1 to n times.

- The ┤ parameters-n &xle. symbol indicates a labelled group that continues below the main syntax diagram. Syntax is occasionally broken into fragments if the inclusion of the fragment would overly complicate the main syntax diagram.

```
►►──KEYword──┤ parameters-1 ├──►◄
```

parameters-1:

```
├──┬─,optional_choice1──────────────────────────────┬──┤
   │                        ┌─,default─────────┐    │
   └─,optional_choice2──────┼──────────────────┼────┘
                            └─,optional_choice─┘
```

Appendix G: Enabling the BatchPipes Dynamic Exit

The BatchPipes dynamic exit is named ASFPEXIT. The BatchPipes dynamic exit uses the IBM dynamic exit service, CSVDYNEX, to do the following:

- Notify authorized programs about the occurrence of certain BatchPipes Subsystem events
- Notify authorized programs about the occurrence of certain BatchPipes Job and Connection events

For example, a program could use the BatchPipes dynamic exit to determine where the actual units of work for a job are executed.

To use the BatchPipes dynamic exit, do the following:

- Create a user exit routine; see "Creating A User Exit Routine"
- Install your user exit routine; see "Installing Your User Exit Routine" on page 292
- Associate your user exit routine with the BatchPipes dynamic exit; see "Associating Your User Exit Routine" on page 293

To become familiar with CSVDYNEX and user exit routines for this service, see the *OS/390 MVS Programming: Authorized Assembler Services Guide* (GC28-1763).

Creating A User Exit Routine

You create a user exit routine to receive and process the information posted by the BatchPipes dynamic exit. Use the information in this section to create your user exit routine.

- See "Exit Routine Environment"
- See "Exit Recovery" on page 290
- See "Program Considerations" on page 290
- See "Register At Entry" on page 290
- See "Return Codes" on page 291
- See "Register At Exit" on page 291
- See "Event Codes" on page 291
- See "Installing Your User Exit Routine" on page 292
- See "Associating Your User Exit Routine" on page 293
- See "Sample User Exit - ASFPASAM" on page 293

Exit Routine Environment

Your user exit routine is given control in the following environment:

Authorization	Supervisor state and PSW key 0
Task or SRB	Task

Cross memory mode
PASN=HASN=SASN

AMODE 31 bit

ASC mode Primary

Interrupt status Enabled for I/O and external interrupts

Locks No locks held

Exit Recovery

If your user exit routine acquires resources, such as storage or locks, provide a recovery routine for the exit. If your user exit routine ends abnormally, control proceeds to the exit's recovery routine or to a system recovery routine. The system recovery routine ends the notification request so that the user exit routine that abended can no longer receive control. BatchPipes continues, however, to send notification to other use exit routines.

Program Considerations

Take the following into consideration when coding your user exit routine:

- Your user exit routine must be reentrant.
- Your user exit routine must be AMODE 31.
- Avoid time-consuming processing such as obtaining large amounts of storage through the GETMAIN macro, issuing a WAIT, issuing an SVC that issues the WAIT macro, or performing I/O operations.
- Since the BatchPipes dynamic exit does not always use the same address space, it cannot depend on the original address space always existing

Register At Entry

On entry to your user exit routine, the indicated registers contain the following information:

Register	Contents
0	Event code (mapped by field ASFPEXIT_QLF in ASFPZEPL macro)
1	Address of a four-word data structure. **Word** **Contents** **1** Address of parameter list (mapped by ASFPEXIT DSECT in ASFPZEPL) **2** Reserved **3** Reserved **4** Reserved
2-12	Do not contain any information for use by the user exit routine
13	Address of an 18-word save area
14	Return address
15	Entry point address of the user exit routine

Return Codes

The user exit routine does not need to set any return codes.

Register At Exit

Upon return from user exit routine, the indicated registers must contain the following information:

Register	Contents
0, 1, and 15	Your user exit routine does not have to place any information in these registers, and does not have to restore its contents to what they were when the user exit routine received control.
2 - 14	Your user exit must restore the contents of these registers to what they were when the exit received control.

Event Codes

The following table lists the event codes for which the BatchPipes dynamic exit supplies information. Each event code is described and the DSECT associated with each event code is listed.

Notes:

1. All event codes are in hexadecimal format.
2. The parameter list is mapped by macro ASFPZEPL.
3. The variable data is pointed to by ASFEXIT_VARIABLE_POINTER.
4. All events for a specific job can be logically grouped together by the token supplied to your user exit routine.

Even Code	Description	Environment	DSECT
1001	The BatchPipes BatchPipes component subsystem is initializing.	BatchPipes component subsystem	ASFPZEPL
1002	The BatchPipes BatchPipes component subsystem is quiescing.	BatchPipes component subsystem address space	ASFPZEPL
1101	BatchPipes Pipe Connection OPEN Requested by user	Pipe Connection address space	ASFPZEPL
1102	BatchPipes Pipe Connection OPEN Completed for user	Pipe Connection address space	ASFPZEPL
1103	BatchPipes Pipe Connection CLOSE Requested by user	Pipe Connection address space	ASFPZEPL
1104	BatchPipes Pipe Connection CLOSE Completed for user	Pipe Connection address space	ASFPZEPL
1105	BatchPipes Pipe Connection Allocation Requested by user	Pipe Connection address space	ASFPZEPL
1106	BatchPipes Pipe Connection Allocation Completed for user	Pipe Connection address space	ASFPZEPL
1107	BatchPipes Pipe Connection Termination Requested by user	Pipe Connection address space	ASFPZEPL
1108	BatchPipes Pipe Connection Termination Completed for user	Pipe Connection address space	ASFPZEPL

Installing Your User Exit Routine

You must install your user exit routine into common storage. Use one of the following methods to ensure that your user exit routine is in common storage:

- Install your user exit routine in LPA either by link-editing the routine into LPA or by using SMP/E.
- Issue either the GETMAIN macro or the STORAGE macro to allocate common storage and copy your user exit routine into the allocated storage.

Associating Your User Exit Routine

You must associate your user exit routine with the BatchPipes dynamic exit. To associate your user exit routine with the BatchPipes dynamic exit, use one of the following:

```
CSVDYNEX REQUEST=ADD statement
MVS SETPROG ADD command
```

To end the association, use one of the following:

```
CSVDYNEX REQUEST=DELETE statement
MVS SETPROG DELETE command
```

For more information on the CSVDYNEX macro service reference the *OS/390 V2R4.0 MVS Authorized Assembler Services Guide*.. For more information on the SETPROG command reference the *MVS/ESA SP V5 System Commands*.

The BatchPipes events are always active, if the subsystem is active. Once the subsystem has been activated, the exit is defined.

Sample User Exit - ASFPASAM

The following example can be used to change the value that is reported as the 'STEP Name' for a pipe connection; this is useful in the case of deciding which step name should be used when the executing program is part of a JCL PROC. In the case of a JCL proc the step name to be used for reporting purposes by BatchPipes could be either the 'step that invoked the proc' -or- the 'step name of the executing program in the PROC'.

This exit code will determine if there is BOTH a 'step name' and a 'proc step name' and, because BatchPipes chooses the former by default, will override that choice and select the ProcStepName instead for the pipe allocation in process. The result of this change is that wherever STEPNAME is reported for this pipe connection by BatchPipes (messages, monitor display SMF records) the ProcStepName value will be used instead of the StepName.

```
         TITLE 'ASFPASAM - BatchPipes Sample ASFPEXIT Exit'
*
ASFPASAM CSECT
ASFPASAM AMODE 31               ADDRESSING MODE
ASFPASAM RMODE ANY              RESIDENCE MODE
*
* *********************************************************************
*                                                                     *
* Module Name: ASFPASAM                                               *
*                                                                     *
* Title: ASFPUTL - BatchPipes Sample ASFPEXIT Sample Exit             *
*                                                                     *
* PROPRIETARY STATEMENT=                                              *
*****PROPRIETARY_STATEMENT*********************************************
*                                                                     *
*  LICENSED MATERIALS - PROPERTY OF IBM                               *
*  THIS PART IS "RESTRICTED MATERIALS OF IBM"                         *
*  5655-D45 (C) COPYRIGHT IBM CORP. 1998, 2000                        *
*                                                                     *
*   STATUS= HACH301                                                   *
*                                                                     *
```

```
*****END_OF_PROPRIETARY_STATEMENT**************************************
*                                                                     *
* Component: Batchpipes/MVS (06500)                                   *
*                                                                     *
* Function: This sample exit is provided by Batchpipes/MVS to use as  *
*           a guide to modifying the ASFPEXIT exit for processing     *
*           BatchPipes events.                                        *
*                                                                     *
* Operation:                                                          *
*                                                                     *
*    - Save input registers and input parameters                      *
*                                                                     *
*    - Determine the event being signalled and process accordingly:   *
*                                                                     *
*      . For a BatchPipes Pipe Allocation Requested Call:             *
*                                                                     *
*          Obtain the current Step and ProcStep Names and if there IS *
*          a Proc Step Name - change the StepName Field in the input  *
*          parameter area to be that value; otherwise do nothing.     *
*                                                                     *
*      . For any other functions:                                     *
*                                                                     *
*          Do Nothing                                                 *
*                                                                     *
*    - Restore the input registers and return to the caller.          *
*                                                                     *
*                                                                     *
* Recovery Operation: None, covered by caller's recovery              *
*                                                                     *
* Notes:                                                              *
*                                                                     *
*   - To prepare this exit for use; the installation must assemble    *
*     and link edit this module into a library (does NOT have to be   *
*     authorized).  On the assembly step the SYSLIB DD statement      *
*     must include the System Macro Library (ie. SYS1.MACLIB) dataset *
*     -and- the BatchPipes Macro Library (ie. SYS1.SASFPMAC) dataset. *
*                                                                     *
*   - The installation must define this module to the ASFPEXIT exit   *
*     Point using the following MVS console command (or similar       *
*     statement in the active PROGxx PARmlib member for the system.   *
*                                                                     *
*      SETPROG EXIT,ADD,EXITNAME=ASFPEXIT,MODNAME=ASFASAM,DSNAME=xxxx *
*                                                                     *
* Dependencies:                                                       *
*                                                                     *
*   - See 'Notes'                                                     *
*                                                                     *
* Character Code Dependencies: EBCDIC                                 *
*                                                                     *
* Restrictions: None                                                  *
*                                                                     *
* Register Conventions: See Register DECLARES section                 *
*                                                                     *
*    Registers-Saved: 0->14                                           *
*                                                                     *
*    Registers-Restored: 0->14                                        *
*                                                                     *
* Module Type: Procedure                                              *
*                                                                     *
* Processor: ASM H                                                    *
*                                                                     *
*   Module Size: See assembled length                                 *
*                                                                     *
*   Attributes: LOCATION      = User's address space                  *
*               STATE         = Supervisor                            *
*               KEY           = 0                                     *
*               MODE          = Task                                  *
*               SERIALIZATION = None                                  *
*               TYPE          = Reentrant                             *
*               AMODE         = 31                                    *
*               RMODE         = Any                                   *
*               HASID         = User's Address Space                  *
*               PASID         = User's Address Space                  *
*               SASID         = User's Address Space                  *
*               ADDR MODE     = Primary Mode                          *
*                                                                     *
* Entry Point:  ASFPASAM .... ASFPEXIT Exit Point                     *
*                                                                     *
*   Linkage: BALR                                                     *
*   Callers: MVS Exit Processing (From BatchPipes)                    *
*   Entry-Registers: 0    = BatchPipes Event Code (see ASFPEXIT_QLF)  *
*                    1    = Address of list of pointers to parmlist   *
*                    2-12 = N/A                                       *
*                    13   = Address of save area                      *
*                    14   = Return Address                            *
*                    15   = Entry Address                             *
*                                                                     *
* Input:                                                              *
*                                                                     *
*   - Register 0 contains the BatchPipes Event Code (see ASFPEXIT_QLF)*
*                                                                     *
*   - Register 1 points to a list of 4 bytes addresses for the        *
*                following parameters:                                *
*        1 - 4 Byte Address of Parameter Area (Mapped by ASFPEXIT)    *
*        2 - 4 Byte Value of '0' (unused)                             *
*        3 - 4 Byte Value of '0' (unused)                             *
*        4 - 4 Byte Value of '0' (unused)                             *
*                                                                     *
* Output:                                                             *
*                                                                     *
*    Depending on the event being signalled the output differs:       *
*                                                                     *
*      . For a BatchPipes Pipe Allocation Requested Call:             *
*                                                                     *
*        = Update StepName value inn the input ASFPZEPL parm area     *
*                                                                     *
* Exit Normal: Return to caller                                       *
*                                                                     *
*   Conditions: Always                                                *
*                                                                     *
*   Exit-Registers: 0-14 - Restored                                   *
*                     15 - Return code                                *
*                                                                     *
*   Return code: 0 - (unused)                                         *
*                                                                     *
* Exits-Error: None                                                   *
*                                                                     *
* External References:                                                *
*                                                                     *
*   Routines: None                                                    *
*                                                                     *
*   Data Areas: None                                                  *
*                                                                     *
*   Control Blocks:                                                   *
*                                                                     *
*      Name       Mapping   Reference                                 *
*      --------   -------   ----------                                *
*      ASFPEXIT   ASFPXEPL  R                                         *
*      ASFPZEPL   ASFPZEPL  R,W                                       *
*      CVT        CVT       R                                         *
*      JESCT      IEFJESCT  R                                         *
*      JSCB       IEZJSCB   R                                         *
*      PSA        IHAPSA    R                                         *
*      SCT        IEFASCTB  R                                         *
*      SWAEPA     IEFZB505  C,R,W,D                                   *
*      TCB        IKJTCB    R                                         *
*                                                                     *
*                           C-CREATED, R-READ, W-WRITTEN, D-DELETED   *
*                                                                     *
* Tables: None                                                        *
*                                                                     *
* Macros-Mapping:                                                     *
*                                                                     *
*    ASFPZEPL ... BatchPipes ASFPEXIT Parameter Area                  *
*    CVT ........ CVT Mapping                                         *
*    IEFASCTB ... SCT Mapping                                         *
*    IEFJESCT ... JESCT Mapping                                       *
*    IEFZB505 ... SWAEPA ParmArea Mapping                             *
*    IEZJSCB .... JSCB Mapping                                        *
*    IHAPSA ..... PSA Mapping                                         *
*    IKJTCB ..... TCB Mapping                                         *
*                                                                     *
* Macros-Executable:                                                  *
*                                                                     *
*    FREEMAIN ... Release virtual storage                             *
*    GETMAIN .... Allocate virtual storage                            *
*    SWAREQ ..... Request address of SWA block (based upon SVA)       *
*                                                                     *
* Serialization: None                                                 *
*                                                                     *
* Messages: None                                                      *
*                                                                     *
* ABEND Codes: None                                                   *
*                                                                     *
* Change Activity:                                                    *
*                                                                     *
* A - Initial level of Code                                       @L0A*
*                                                                     *
* $MOD(ASFPASAM) COMP(06500) PROD(HACH201): Sample ASFPEXIT Exit      *
*                                                                     *
* Flag LineItem  FMID   DATE   ID   Comment                           *
*                                                                     *
* $00= PQ18930 HACH201 981005  WRR: Dynamic STEPName Reporting    @00A*
*                                                                     *
* **********************************************************************
*
         EJECT
*
*---------------------------------------------------------------------
* Initialize module
*---------------------------------------------------------------------
*
         USING ASFPASAM,INEPA
         B     START                    Branch around MOD ID Data
         DC    CL8'ASFPASAM'            Module Name
         DC    CL8'&SYSDATE'            Module Assembly Date
         DC    CL8'&SYSTIME'            Module Assembly Time
START    STM   R14,R12,R14INSA(INPUTSA) Store regs in caller's savearea
         DROP  INEPA
         LR    CODEREG,INEPA            Establish Base Reg
         USING ASFPASAM,CODEREG
*
*---------------------------------------------------------------------
* Save input parameters and initialize global variables
*---------------------------------------------------------------------
*
         L     EXITPTR,0(R1)            Get Input ASFPEXIT Address
         USING ASFPEXIT,EXITPTR
         L     EPLPTR,ASFPEXIT_VARIABLE_POINTER Get ASFZEPL Address
         USING ASFPZEPL,EPLPTR
*
*---------------------------------------------------------------------
* Initialize local variables
*---------------------------------------------------------------------
*
         SLR   WAPTR,WAPTR              Init WorkArea Pointer
*
*---------------------------------------------------------------------
* Initialize pointers to global control blocks (PSA, TCB)
*---------------------------------------------------------------------
*
         USING PSA,0                    Set PSA Basing
         L     TCBPTR,PSATOLD           Get TCB address
         DROP  0                        Release PSA Basing
         USING TCB,TCBPTR               Set TCB Basing
         L     JSCBPTR,TCBJSCB          Get JSCB Address
         USING IEZJSCB,JSCBPTR          Set JSCB Basing
         L     JSCBPTR,JSCBACT          Get JSCB Address
*
*---------------------------------------------------------------------
* Check input event code and process, if needed
*---------------------------------------------------------------------
*
         CL    INCODE,=AL4(ASFPEXIT_QLF_BP_PIPEALLOC_REQ) Check Event
         BNE   EXIT                     No Match - Terminate Processing
*
*---------------------------------------------------------------------
* Obtain the local work area
*---------------------------------------------------------------------
*
         GETMAIN RU,LV=WALEN,SP=WASPL,LOC=ANY   Get work area
         LR    WAPTR,R1                 Save address of workarea
         USING WORKAREA,WAPTR           WORKAREA DSECT Basing
         XC    WORKAREA,WORKAREA        Init Storage Area
*
*---------------------------------------------------------------------
* Invoke SWAREQ to obtain the actual address of the SCT
*---------------------------------------------------------------------
*
         LA    ZB505PTR,ZB505WA         Get Pointer to Storage
         USING ZB505,ZB505PTR           Set WorkArea Basing
*
         MVC    SWVA,JSCSCTP            Get SCT SVA
         ST     R13,R13SAVE             Save SaveArea Address
         LA     R13,SAVEAREA            Set SaveArea Address
         ST     ZB505PTR,SWAEPPTR       Save EPA Address
         SWAREQ FCODE=RL,EPA=SWAEPPTR,MF=(E,LSWAREQ)
         L      R13,R13SAVE             Restore SaveArea Address
         LTR    R15,R15
         BNZ    EXIT
         L      SCTPTR,SWBLKPTR         Get SCT Address
         USING  SCT,SCTPTR              Set SCT Basing
*
*---------------------------------------------------------------------
* After obtaining the SCT, check for a PROCSTEPName - if present
* change the input step name to be used by BatchPipes (for reporting
* purposes)
*---------------------------------------------------------------------
*
         CLC   SCTSCLPC,=CL8'        '  PROC StepName ???
         BE    EXIT                     No - bypass processing
         MVC   ASFPZEPL_STEPNAME,SCTSCLPC Use PROCSTEP Name
*
*---------------------------------------------------------------------
* Terminate processing and return to the caller
*---------------------------------------------------------------------
*
EXIT     EQU   *
*
*---------------------------------------------------------------------
* Check if workarea has been allocated; if so free it.
*---------------------------------------------------------------------
*
         LTR   WAPTR,WAPTR              Check WorkArea Pointer
         BZ    NOWAFREE                 Skip WorkArea Free
         FREEMAIN   RU,LV=WALEN,A=(WAPTR),SP=WASPL   Free workarea
*
*---------------------------------------------------------------------
* Restore input regs and return to caller
*---------------------------------------------------------------------
*
NOWAFREE EQU   *
         LM    R14,R12,R14INSA(INPUTSA) Restore inout regs
         SLR   R15,R15                  Set RC = 0
         BR    R14                      Return to caller
*
         EJECT
*
*-- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --*
*   Define Registers                                                  *
```

```
*-- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --*
*
R0       EQU   00                      REG 0
INCODE   EQU   00                      Input Event Code
R1       EQU   01                      REG 1
EXITPTR  EQU   02                      Pointer to input ASFPEXIT Area
EPLPTR   EQU   03                      Pointer to input ASFPZEPL Area
WORK     EQU   04                      Temp Hold Area
ZB505PTR EQU   06                      Pointer to ZB505 Work Area
SCTPTR   EQU   07                      Pointer to SCT
JSCBPTR  EQU   08                      Pointer to JSCB
TCBPTR   EQU   09                      Pointer to TCB
WAPTR    EQU   11                      Pointer to local workarea
CODEREG  EQU   12                      Module Code Register
R12      EQU   12                      REG 12
R13      EQU   13                      Pointer to local savearea
INPUTSA  EQU   13                      Pointer to caller's savearea
R14      EQU   14                      REG 14
INEPA    EQU   15                      Input Module Address
R15      EQU   15                      REG 15
*
*-- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --*
*   Define work constants                                             *
*-- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --*
*
R01INSA  EQU   24                      Offset of Reg '1' in SaveArea
R14INSA  EQU   12                      Offset of Reg '14' in SaveArea
R15INSA  EQU   16                      Offset of Reg '15' in SaveArea
*
*-- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --*
*
         EJECT
*
*-- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --*
*   Define data constants and macro parmlist template                 *
*-- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --*
*
*
         EJECT
*
*-- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --*
*   Define local variables                                            *
*-- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --*
*
WASPL    EQU   230                   WorkArea subpool
WORKAREA DSECT                       Local Work Area
         DS    0D                    Init Alignment
SAVEAREA DS    18F                   Register savearea
         DS    0D
LSWAREQ  SWAREQ FCODE=RL,EPA=ZB505WA,MF=L
         DS    0D
R13SAVE  DS    A
SWAEPPTR DS    A
ZB505WA  DS    0D
         ORG   *+ZB505LEN
         DS    0D
WALEN    EQU   *-WORKAREA            Length of work area DSECT
*
```

```
         EJECT
*
*-- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --*
*   Include control block mappings                                    *
*-- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --*
*
*
         ASFPZEPL                    BatchPipes ASFPEXIT ParmArea
*
         EJECT
*
         IKJTCB                      TCB mapping
*
         EJECT
*
         IEZJSCB                     JSCB mapping
*
         EJECT
*
         DS    0D
SCT      DSECT
         IEFASCTB                    SCT mapping
*
         EJECT
*
         CVT DSECT=YES               CVT mapping
*
         EJECT
*
         IHAPSA                      PSA mapping
*
         EJECT
*
         IEFJESCT                    JESCT mapping
*
         EJECT
*
         IEFZB505                    SWAREQ Parms
*
ZB505LEN EQU   *-ZB505
*
         EJECT
*
*
         EJECT
*
         END   ASFPASAM              End of exit module definition
```

Glossary

Glossary terms are defined as they are used in this book. If you can't find the term you are looking for, refer to the index or to the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

Closed connection. A job/pipe connection that is deleted due to the job issuing the CLOSE macro.

Connection to the pipe. A job/pipe association that occurs when a job issues an OPEN macro for a data set that is defined to be handled by BatchPipes. A pipe has as many connections as issuances of the OPEN macro for the data set that is a pipe.

Consumption rate. The rate at which a reader is able to READ/GET records or blocks over a certain span of time.

Critical path. That group of jobs that run together to attain a defined goal. Additionally, the jobs must depend on each other so that a delay in any one of the jobs causes a delay of the entire workload.

Destructive read. Once a record or a block has been read from the pipe, the data is no longer available to be read by any other job.

Disconnection from the pipe. The ending of the job/pipe association. It occurs when a job issues a CLOSE request for the data set that is defined as a pipe.

Empty pipe. A pipe that has no data in it. If a reader issues a GET for data, the reader will wait until data is written to the pipe or a CLOSE is issued by all the writers.

Filter. A job (reader or writer) that you add to a pipeline to change the way data flows in the pipeline; some examples are:

- A process to exclude records from a pipe
- A sort
- Data hardening

Full pipe. A pipe that has no room for more records. If a writer issues a PUT when the pipe is full, the writer will wait until a reader issues a GET.

Hardening data. The action required to create a physical backup of a “piped” data set. It involves having a job write to the pipe and also write to tape or DASD.

Idle time. Time during which a pipe connection has no outstanding write or read requests. For example, a writer that is waiting for a tape mount is not using the pipe; it is idle.

Inactivity threshold. The period of time that an installation establishes for idle, waiting, or waiting-for-open jobs. When a job reaches this time limit, BatchPipes sends a message to a system console stating that the job has exceeded the inactivity threshold.

Inactivity threshold exception. A job that has been waiting, idle, waiting-for-open for a period of time that exceeds the installation's setting for an inactivity threshold.

Many-to-one pipeline. A pipeline where more than one writer is writing to a pipe from which one reader is reading.

One-to-many pipeline. A pipeline where one writer is writing to a pipe from which more than one reader is reading.

Opened connection. A job/pipe connection that BatchPipes builds when a job using pipes issues the OPEN macro.

Partner. A job at the other pipe connection. For a writer, a reader from the pipe is a partner; for the reader, a writer to the pipe is a partner. A job can have more than one partner.

Pipe. A virtual storage buffer — a FIFO data-in-memory queue through which BatchPipes transfers data from writer to reader.

Pipe balance. The ratio of records-being-written to records-being-read over an interval of time. The pipe is balanced if the ratio approximates one and both the reader and the writer spend very little time waiting to use the pipe. The pipe is not balanced if the record production and consumption rates are considerably different. In this case one of the partners spends time waiting for the other to write or read data.

Pipe depth. The number of records or blocks permitted in the pipe before the pipe is full.

Pipe name. A data set name that identifies a pipe.

Pipeline. A job-pipe-job association. One or more pipes with one or more writer/reader partners connected. A writer-pipe-reader configuration represents a basic pipeline.

Production rate. The rate at which a writer is able to WRITE/PUT records/data over elapsed time.

Read connection. A job/pipe association that occurs when a job issues an OPEN macro for a data set to be handled by BatchPipes and reads from that data set.

Reader. A job that reads (through BSAM READ or QSAM GET) data from a pipe.

Recovery boundary. A point in a jobstream where you harden data, from which you can restart the jobs in the case of an abnormal ending of the jobs in a pipeline.

Stage. The parts of a pipeline that represent the processing elements, or jobs. A basic pipeline consists of writer-pipe-reader partners. This is a two stage pipeline. The writer provides a particular function at the first stage, and the reader a different function at the second stage. There can be many writers and/or many readers at a particular stage in the pipeline.

Subsystem recording interval. A time interval, set through the SMFPRMxx parmlib member, that determines how often BatchPipes writes Type 91 Subsystem Interval Records.

Suite of jobs. All the jobs associated with a specific business process. A suite of jobs can include jobs using BatchPipes.

Threshold. See “inactivity threshold.”

Topology. A network of pipes and jobs using those pipes.

Wait time. Time spent by a writer waiting on a full pipe, or by a reader waiting on an empty pipe. BatchPipes keeps track of the total wait times and of each instance of waiting.

Wait-for-open. The condition that exists when one job has issued an OPEN for a pipe, but no partner job has issued an OPEN for the same pipe. Until the job and a partner both issue OPENs, the pipe does not exist.

Writer. A job that writes (BSAM WRITE, QSAM PUT) data to a pipe.

Write connection. A job/pipe association that occurs when a job issues an OPEN macro for a data set to be handled by BatchPipes and writes to that data set.

Writer/reader partners. Jobs that use the same pipe such that data transfers from one to another. At least one writer/reader pair must exist for a connection to occur.

Index

Communicating Your Comments to IBM

IBM BatchPipes OS/390 V2R1
Users Guide and Reference

Publication No. SA22-7458-00

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM 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.

If you are mailing a reader's comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:
  - FAX: (International Access Code)+1+845+432-9405
- If you prefer to send comments electronically, use one of these network IDs:
  - IBM Mail Exchange: USIB6TC9 at IBMMAIL
  - Internet e-mail: mhvrcfs@us.ibm.com
  - World Wide Web: http://www.ibm.com/s390/os390/

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies

Optionally, if you include your telephone number, we will be able to respond to your comments by phone.

Reader's Comments — We'd Like to Hear from You

IBM BatchPipes OS/390 V2R1
Users Guide and Reference

Publication No. SA22-7458-00

You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Today's date: ________________________________

What is your occupation?

Newsletter number of latest Technical Newsletter (if any) concerning this publication:

How did you use this publication?

[ ]	As an introduction	[ ]	As a text (student)
[ ]	As a reference manual	[ ]	As a text (instructor)
[ ]	For another purpose (explain)		

____

____

Is there anything you especially like or dislike about the organization, presentation, or writing in this manual? Helpful comments include general usefulness of the book; possible additions, deletions, and clarifications; specific errors and omissions.

Page Number: Comment:

______________________________ ______________________________
Name Address

______________________________ ______________________________
Company or Organization

______________________________ ______________________________
Phone No.

Reader's Comments — We'd Like to Hear from You
SA22-7458-00

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
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400

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

Cut or Fold
Along Line

File Number: S370/S390-34
Program Number: 5655-D45

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SA22-7458-00