z/VM
7.3

OpenExtensions Callable Services Reference

SC24-6296-73

Note:

Before you use this information and the product it supports, read the information in "Notices" on page 563.

This edition applies to version 7, release 3 of IBM® z/VM® (product number 5741-A09) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2022-08-31

© Copyright International Business Machines Corporation 1993, 2022.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures

Tables

About This Document

This document describes the IBM z/VM OpenExtensions callable services and the mapping macros related to the callable services. These services are interfaces between the z/VM operating system and the functions specified in the POSIX.1 standard (ISO/IEC 9945-1:1990[E] IEEE Std 1003.1-1990: First edition 1990-12-07; Information Technology—Portable Operating System Interface [POSIX] Part 1; System Application Program Interface [API] [C Language]). These functions are used by z/VM POSIX support. This document also describes callable services that are not specified in the standards.

Intended Audience

This information is for assembler programmers who want to use the z/VM POSIX support interface.

Where to Find More Information

More detailed information on the z/VM POSIX support can be found in the following documents:

- *z/VM: OpenExtensions POSIX Conformance Document*
- *z/VM: OpenExtensions User's Guide*
- *z/VM: OpenExtensions Commands Reference*

Other documents you might need to develop application programs are listed in the "Bibliography" on page 567.

Using the Online HELP Facility

You can receive online information about the OpenExtensions callable services and macros described in this book by using the VM HELP Facility. For example, to display a menu of OpenExtensions callable services, enter:

```
help oroutine menu
```

To display a menu of OpenExtensions macros, enter:

```
help omacro menu
```

To display information about a specific OpenExtensions service (such as access, BPX1ACC), enter one of the following commands:

```
help oroutine access
help oroutine bpx1acc
```

Because of the length of some of the routine names, typing the first eight characters of a routine's name (omitting the underscores) may not provide help for the desired routine. For example, entering:

```
help oroutine pthreadc
```

could mean you would like help for pthread_cancel or pthread_create. In this case, you can try an abbreviation for the routine name. For example, to request help on pthread_cancel, enter:

```
help oroutine pthcance
```

Of course, you can always request help on a routine by using its BPX name. To request help on pthread_create (BPX1PTC), enter:

```
help oroutine bpx1ptc
```

© Copyright IBM Corp. 1993, 2022

For more information about using the HELP Facility, see the *z/VM: CMS User's Guide*. To display the main HELP Task Menu, enter:

```
help
```

For more information about the HELP command, see the *z/VM: CMS Commands and Utilities Reference* or enter:

```
help cms help
```

Links to Other Documents and Websites

The PDF version of this document contains links to other documents and websites. A link from this document to another document works only when both documents are in the same directory or database, and a link to a website works only if you have access to the Internet. A document link is to a specific edition. If a new edition of a linked document has been published since the publication of this document, the linked document might not be the latest edition.

How to Send Your Comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or give us any other feedback that you might have.

To send us your comments, go to z/VM Reader's Comment Form (https://www.ibm.com/systems/campaignmail/z/zvm/zvm-comments) and complete the form.

If You Have a Technical Problem

Do not use the feedback method. Instead, do one of the following:

- Contact your IBM service representative.
- Contact IBM technical support.
- See IBM: z/VM Support Resources (https://www.ibm.com/vm/service).
- Go to IBM Support Portal (https://www.ibm.com/support/entry/portal/Overview).

© Copyright IBM Corp. 1993, 2022

Summary of Changes for z/VM: OpenExtensions Callable Services Reference

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line (**|**) to the left of the change.

SC24-6296-73, z/VM 7.3 (September 2022)

This edition supports the general availability of z/VM 7.3. Note that the publication number suffix (-73) indicates the z/VM release to which this edition applies.

SC24-6296-01, z/VM 7.2 (July 2021)

This edition includes terminology, maintenance, and editorial changes.

SC24-6296-01, z/VM 7.2 (March 2021)

This edition includes terminology, maintenance, and editorial changes.

SC24-6296-01, z/VM 7.2 (September 2020)

This edition supports the general availability of z/VM 7.2.

SC24-6296-00, z/VM 7.1 (September 2018)

This edition supports the general availability of z/VM 7.1.

© Copyright IBM Corp. 1993, 2022

Chapter 1. Invocation Details for Callable Services

As an interface between the z/VM operating system and the functions specified in the POSIX.1 standard, OpenExtensions provides access to a set of assembler callable services known as the OpenExtensions callable services. These callable services have a standard set of syntax and linkage requirements as well as parameter specification details necessary for successful invocation.

Establishing the OpenExtensions Environment

The OpenExtensions callable services are provided to enable language run-time environments to implement the POSIX interface and to provide system programmers access to a language-neutral subset of the POSIX functions. Because this is essentially an interface for the writer of a language run-time environment, the following guidelines apply to its use.

The create_thread_environment (BPX1CTE) service should be called before calling any of the other OpenExtensions callable services. The create_thread_environment (BPX1CTE) service causes the initialization of the POSIX process environment in the CMS session, including the establishment of a caller-specified language environment manager to handle language-specific threading conditions.

If any other OpenExtensions callable service is called before create_thread_environment (BPX1CTE), POSIX process initialization is implicitly performed, but the default assembler language environment manager is established for the process.

Syntax Conventions for the Callable Services

A callable service is a programming interface that uses the CALL macro to access system services. To code a callable service, code the CALL macro followed by the name of the callable service and a parameter list. A syntax diagram for a callable service follows.

```
function_name
    parm_1
    parm_2
    .
    .
    return_value
    return_code
    reason_code
```

This format does not show the assembler column dependance (columns 1, 10, 16, and 72) or parameter list options (VL and MF). The exact syntax is shown in the examples in “Reentrant Entry Linkage” on page 551.

Considerations for coding callable services are:

- You must code all the parameters in the parameter list because parameters are positional in a callable service interface. That is, the function of each parameter is determined by its position with respect to the other parameters in the list. Omitting a parameter, therefore, assigns the omitted parameter's function to the next parameter in the list.
- You must place values explicitly into all supplied parameters, because callable services do not set defaults.

Function_Name
The name that assembler understands is the name of an entry point in the form BPX1*xxx*, where *xxx* is a three-character symbol unique to the service.

© Copyright IBM Corp. 1993, 2022

This entry point is a stub routine, bound into your program at module build time, for a CMS Callable Services Library (CSL) routine.

Parm Parameters
The parameters *parm_1*, *parm_2*, and so on, are placeholders for variables that may be part of a service's syntax.

Return_Value
The *return_value* parameter is a common parameter to many callable services. It indicates the success or failure of the service. If the callable service fails, it returns a -1 in the *return_value*. For most successful calls to OpenExtensions services, the return value is set to 0. However, some services, such as "getgrgid (BPX1GGI) — Access the Group Database by ID" on page 117 and "getgrnam (BPX1GGN) — Access the Group Database by Name" on page 119, return zeros instead of -1 when the service fails.

Some callable services, such as spawn (BPX1SPN), return a positive return value to indicate success. Other services are unique, such as "_exit (BPX1EXI) — End a Process and Bypass the Cleanup" on page 79, in that they do not return when successful.

Some services do not have a return value, because the services do not fail under normal conditions. System failures, however, may cause those services to fail. In this case, the process that issues the call abends. See "getegid (BPX1GEG) — Get the Effective Group ID" on page 114 for an example.

Return_Code
The *return_code* parameter is referred to as the **errno** in the POSIX C interface. The *return_code* is returned only if the service fails.

In the callable service description, some of the possible return codes are listed for services that have return codes. The return codes are described in each service if they help describe its function.

Reason codes are listed with the return code that they describe.

All the return codes and their descriptions are found in Appendix A, "Return Codes," on page 487.

Some *return_code* values may occur for any callable service: the OpenExtensions unique return codes. They are not always listed under each callable service. See Appendix A, "Return Codes," on page 487 for a description of each of these.

Reason_Code
The *reason_code* parameter usually accompanies the *return_code* value when the callable service fails. It further defines the return code. Reason codes do not have a POSIX equivalent.

All the reason codes and their descriptions are found in Appendix B, "Reason Codes," on page 495. Reason codes are listed both alphabetically by name and numerically by value. The value is the lower half of the reason code.

Linkage Conventions for the Callable Services

Callers must use the following linkage conventions for all OpenExtensions callable services:

- Register 1 is set up by the CALL macro with the address of a parameter list, which is a list of consecutive words, each containing the address of a parameter to be passed. The last word in this list must have a 1 in the high-order (sign) bit.
- Register 14 is set up by the CALL macro; it contains the return address.
- Register 15 is set up by the CALL macro; it contains the entry point address of the service stub being called.

The OpenExtensions callable services do not use the contents of any registers other than 1, 14, and 15.

Programming Language Binding Files

CMS provides language binding files that define function entry points and constants used by the OpenExtensions callable services.

VMASMOVM MACRO is the binding file for Assembler. It contains definitions for the VMPOSGNL system event and includes the macros listed in Table 1 on page 3.

Table 1. OpenExtensions Assembler Macros

Macro	Function
BPXYACC	Maps flag values for the access (BPX1ACC) service
BPXYAUDT	Maps flag values for the chaudit (BPX1CHA) and fchaudit (BPX1FCA) services
BPXYBRLK	Maps the byte range lock request for the fcntl (BPX1FCT) service
BPXYCID	Maps the response structure for the getclientid (BPX1GCL) service
BPXYCONS	Defines constants used by OpenExtensions services
BPXYCW	Defines serialization constants used by OpenExtensions services
BPXYDIRE	Maps directory entries for the readdir (BPX1RDD) service
BPXYERNO	Defines component return and reason codes
BPXYFCTL	Maps command values and flags for the fcntl (BPX1FCT) service
BPXYFTYP	Defines file types
BPXYGIDN	Maps data returned for the getpwnam (BPX1GPN) and getpwuid (BPX1GPU) services
BPXYGIDS	Maps data returned for the getgrnam (BPX1GGN) and getgrgid (BPX1GGI) services
BPXYINHE	Maps the spawn (BPX1SPN) inheritance structure
BPXYIOCC	Maps command constants for the w_ioctl (BPX1IOC) service
BPXYIOV	Maps the I/O vector structure used by the readv (BPX1RDV), writev (BPX1WRV), sendmsg (BPX2SMS), and recvmsg (BPX2RMS) services
BPXYIPCP	Maps interprocess communications permissions
BPXYIPCQ	Maps the w_getipc (BPX1GET) data structure
BPXYMNT	Maps the modes for the mount (BPX1MNT) service
BPXYMODE	Maps the mode constants of the file services
BPXYMSG	Maps interprocess communications message queues
BPXYMSGF	Maps the message flags used by the send (BPX1SND), recv (BPX1RCV), sendmsg (BPX2SMS), and recvmsg (BPX2RMS) services
BPXYMSGH	Maps the message header used by the sendmsg (BPX2SMS) and recvmsg (BPX2RMS) services
BPXYMTM	Maps the modes for the mount (BPX1MNT) and umount (BPX1UMT) services
BPXYOPNF	Maps flag values for the fcntl (BPX1FCT) and open (BPX1OPN) services
BPXYPCF	Defines command values for the pathconf (BPX1PCF) and fpathconf (BPX1FPC) services
BPXYPGPS	Maps the response structure for the w_getpsent (BPX1GPS) service
BPXYPPSD	Maps signal delivery data
BPXYPTAT	Maps attributes for the pthread services
BPXYPTXL	Maps the parameter list for the pthread services

Table 1. OpenExtensions Assembler Macros (continued)

Macro	Function
BPXYSEEK	Defines constants for the lseek (BPX1LSK) service
BPXYSEL	Maps data structures and constants for the select/selectex (BPX1SEL) service
BPXYSELT	Maps the timeout value for the select/selectex (BPX1SEL) service
BPXYSEM	Maps interprocess communications semaphores
BPXYSHM	Maps interprocess communications shared memory segments
BPXYSIGH	Defines signal constants
BPXYSINF	Maps the wait-extension (BPX1WTE) Siginfo_t structure
BPXYSOCK	Maps the SOCKADDR data structure and constants used by socket-related services
BPXYSSTF	Maps the response structure for the fstatvfs (BPX1FTV), statvfs (BPX1STV), and w_statvfs (BPX1STF) services
BPXYSTAT	Maps the response structure for the stat (BPX1STA) service
BPXYTIMS	Maps the response structure for the times (BPX1TIM) service
BPXYTIOS	Maps the termios structure
BPXYUTSN	Maps the response structure for the uname (BPX1UNA) service
BPXYVM5	Defines function code values for the openvmf (BPX1VM5) service
BPXYVM6	Defines function code values for the setopen (BPX1VM6) service
BPXYVM7	Defines function code values for the openvmf7 (BPX1VM7) service
BPXYWAST	Maps the wait status word

VMREXOVM COPY is the binding file for REXX. It includes the definitions for the VMPOSGNL system event as well as the constants defined by the macros listed for VMASMOVM MACRO. However, no equivalent to the DSECT mappings defined by those macros is provided in VMREXOVM. REXX applications should use the `parse` and `substr` instructions to interpret the contents of buffers returned by OpenExtensions callable services.

The **VMCOVM H** file is also provided for C. It contains definitions for the VMPOSGNL system event, but it does not include bindings for the OpenExtensions callable services. The POSIX bindings can be used for C applications.

Invocation from REXX Procedures

Callers from REXX must use the REXX mechanism for calling routines from a Callable Services Library (CSL). A subcommand environment called OPENVM is provided to make invocation of these callable services look like other requests for host functions. After addressing the OPENVM subcommand environment, the services are invoked by specifying the routine name followed by the parameters. In addition, the OPENVM language binding file that defines REXX variables used by the OpenExtensions services should be included by using the APILOAD function. An example of a REXX invocation of one of the callable services follows.

```
trace R  /* *** Show results of each command.  *** */
call apiload 'VMREXOVM'
/* Change the working directory */
pathname = '/home/myfiles'
plength = length(pathname)
address OPENVM
```

```
  'BPX1CHD plength pathname return_value' ,
               'return_code reason_code'
  say return_value  /* *** Show what happened.  *** */
```

Parameter Descriptions for Callable Services

All the parameters of the OpenExtensions callable services described in this book are **required** positional parameters. When you call the service, you must specify all the parameters in the order listed.

Note: Some parameters do not require values and allow you to substitute zeros for the parameter. The descriptions of the parameters identify those that can be replaced by zeros, and when to do so.

The description of each parameter begins with the three-part notation:

(*usage,type,length*)

In this notation:

usage
is one of the following, indicating how the variable is used by the called function:

input
You must supply a value for the parameter in the call.

output
The service returns a value in the parameter when the call is finished.

input/output
The same parameter is used to supply a value to the service and return a value from the service.

type
is one of the following, indicating the type of data the parameter contains:

INT
Signed binary integer

CHAR
Character string

PTR
Pointer to the data described by the next parameter

length
is the length of the variable, specified as one of the following:

- The number of bytes or characters (depending on the data type)
- The number of equal-length elements in an array
- The name of another parameter that specifies the number of bytes, characters, or elements.

Call Parameter Lists

Every callable service is called with a parameter list. As shown in Figure 1 on page 6, when a service is called:

- Register 1 points to a parameter address list.
- Each field in the parameter address list points to a field containing a parameter.
- The "parameter list" is the set of those parameters, however they are arranged in storage. The last parameter pointer in the list must have the high-order bit set to 1.

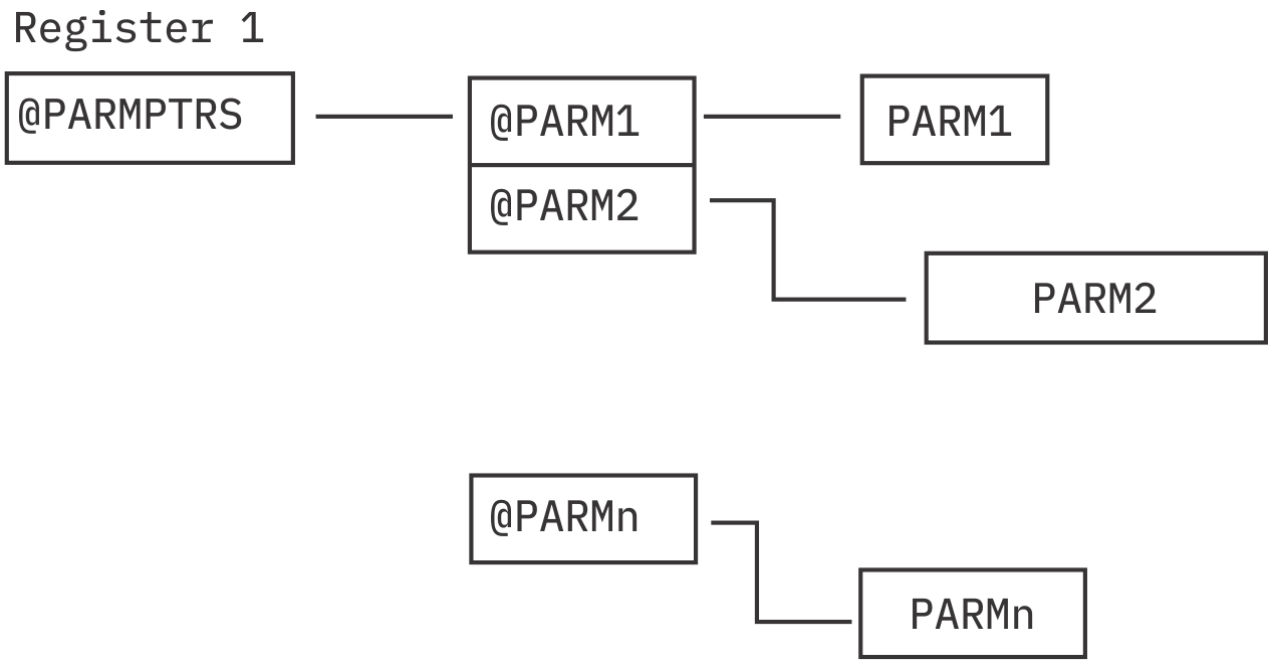


Figure 1. Call Parameter List

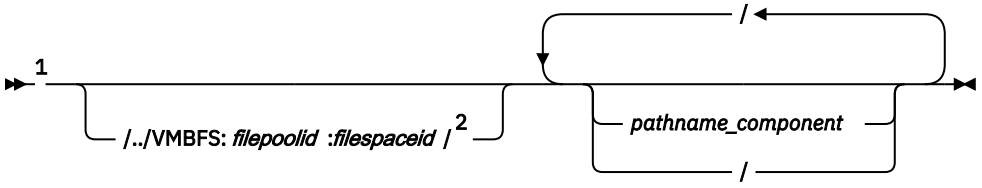
Understanding Byte File System (BFS) Path Name Syntax

All OpenExtensions Byte File System (BFS) objects (files, directories, and so on) are identified through path names. A path name has an optional beginning slash, followed by one or more path name components separated by slashes. A path name component is a string of characters used to identify a BFS object.

A BFS path name may represent a file system accessed through the Network File System (NFS). The NFS file system may be on a remote or local system, which may be VM or non-VM. The mount (BPX1MNT) service or the OPENVM MOUNT command links an NFS file system to a BFS path name, enabling it to be used on most commands and interfaces that accept BFS path names.

The BFS path name identifier is shown as one word, *pathname*, when it depicts a specific path name variable.

Format



Notes:

[1] The minimum path name is a single slash (/).
[2] The ending slash is required only if one or more path name components are also specified.

Parameters

/../VMBFS
is a keyword string that indicates a **fully qualified VM byte file system root**, which identifies the byte file system in which the specified object resides. The VMBFS keyword is not case sensitive.

: (colon)
is a separator that must be specified following the VMBFS keyword and the *filepoolid*.

filepoolid
is the name of the file pool that contains the byte file system. The file pool name can be up to eight characters long. The first character must be alphabetic, but the remaining characters can be alphabetic or numeric. This name is not case sensitive.

filespaceid
is the name of the file space in which the byte file system resides. The file space ID can be up to eight characters long. This name is not case sensitive.

/ (slash)
is a separator that must be specified after the *filespaceid* if path name components are specified. The slash must also be specified between path name components.

When / is specified as a single-character path name, it indicates the root (top) directory of the currently-mounted byte file system.

pathname_component
is the name of an object in the BFS hierarchy. A path name component may be 1-255 characters in length. The slash character (/) and the null character (X'00') are not valid within a path name component. If multiple path name components are specified, they must be separated by slashes. All path name components prior to the last one specified are interpreted as directory names in the hierarchy. The last path name component, if not followed by a slash, may or may not be a directory. When the last path name component is followed by a slash, it is always interpreted as a directory.

Path name component names are case sensitive.

Usage Notes

1. A byte file system may be enrolled in the same file pool as other byte file systems and SFS users.
2. In the OpenExtensions environment, all byte file systems are uniquely identified with the /../vmbfs:*filepoolid*:*filespaceid* construct.
3. Path names can be specified in several ways:
   - When the first character of the path name is not a slash, the path name is known as a **relative path name**. The search for the file starts at the working directory. To establish the working directory, use the chdir (BPX1CHD) service or the OPENVM SET DIRECTORY command. To find the value of the current working directory, use the getcwd (BPX1GCW) service or the OPENVM QUERY DIRECTORY command.
   - When /../vmbfs:*filepoolid*:*filespaceid* is specified at the start of a path name, it is referred to as a **fully qualified path name**. The file is searched for in the byte file system, which is defined as file space *filespaceid* in file pool *filepoolid*. The byte file system does not need to be explicitly mounted.
   - When the path name starts with a slash (but not /../vmbfs:*filepoolid*:*filespaceid*), the path name is known as an **absolute path name**. The search for the file starts from the root of the currently mounted byte file system. The root directory can be established by using the mount (BPX1MNT) service or the OPENVM MOUNT command, or by the POSIXINFO FSROOT statement in your CP directory entry. To find the value of the root directory, use the uname (BPX1UNA) service or the OPENVM QUERY MOUNT command.

   See *z/VM: OpenExtensions Commands Reference* for more information on OPENVM commands.
4. The entire path name must be in the range of 1-1023 characters. Individual path name components cannot exceed 255 characters. All characters are valid within a path name, with the following restrictions:
   - The null character (X'00') is not permitted within a path name.
   - A slash (/) is interpreted as the delineator of a path name component.

   For an application to be portable to the broadest set of environments, POSIX standards suggest that the application restrict the path name as follows:

- Do not exceed 14 characters for any path name component.
- Use only these characters:

  A-Z
  Uppercase alphabetic

  a-z
  Lowercase alphabetic

  0-9
  Numeric

  .
  Period

  _
  Underscore

  -
  Dash

5. Path name components are case sensitive. Note that Abc, abC, and ABC are valid unique path name components.
6. Specifying a path name that begins with exactly two slashes (//) is not permitted; the request will be rejected. A path name that begins with a single slash or three or more slashes is accepted.
7. There are two path name components (file names) that have special meaning during path name resolution. These are:

   .
   The path name component consisting of the single dot character (`.`). When dot is encountered in the path name, it refers to the directory specified by the preceding path name component.

   Some dot (.) examples:

   a. If you specified a path name of:

   ```
   /joes/recipes/./pie
   ```

   It would be equivalent to:

   ```
   /joes/recipes/pie
   ```

   b. If you specified a path name of:

   ```
   ./joes
   ```

   It would be equivalent to:

   ```
   joes
   ```

   ..
   The path name component consisting of two dot characters (`..`). When dot dot is encountered in the path name, it refers to the parent directory of its predecessor. As a special case, in the root directory, dot dot refers to the root directory itself. The construct `/../vmbfs:`*filepoolid*`:`*filespaceid*, as described above, is the only exception.

   Some dot dot (..) examples:

   a. If you had previously set your working directory (using chdir or OPENVM SET DIRECTORY) to:

   ```
   /joes/recipes/
   ```

   and you specified a relative path name of `../tools`, this would be equivalent to specifying an absolute path name of:

   ```
   /joes/tools
   ```

b. If you are working in `/bin/util/src`, and you want to go to `/bin/util`, you can enter:

```
openvm set directory ..
```

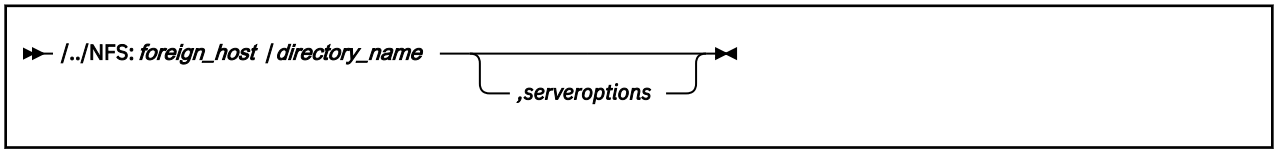
c. If you are working in `/u/rexx/prog/src`, and you want to refer to the file `test` in the directory `/u/rexx/appl/examples`, you could use the following path name to refer to that file:

```
../../appl/examples/test
```

Understanding Network File System (NFS) Path Name Syntax

The Network File System (NFS) path name identifies a file system or directory accessed through NFS. It may be on a remote or local system, which may be VM or non-VM.

Format



Parameters

/../NFS
is a keyword string that indicates the specified path name is a fully-qualified remote file system, accessed by way of a Network File System server. The NFS keyword is not case sensitive.

: (colon)
is a separator that must be specified following the NFS keyword.

foreign_host
identifies the name of the foreign host. Specify *foreign_host* using an internet host name or a dotted-decimal address. This name is not case sensitive.

/ (slash)
is a separator that must be specified following the *foreign_host*.

directory_name
identifies the file system or directory to be mounted. The format of *directory_name* is dependent upon the operating system running at the site identified by *foreign_host*. This name may be case sensitive.

serveroptions
are NFS server MOUNT options, which depend upon the NFS server at *foreign_host*.

The delimiter between *directory_name* and *serveroptions* is defined by the remote host. Typically a comma is used.

Unexpected results may occur if the user name, UID, or GID information you specify in *serveroptions* is not consistent with what the NFS client uses. See the `NETRC`, `USERID`, and `ANONYMOUS` parameters of the OPENVM MOUNT command in the *z/VM: OpenExtensions Commands Reference* for information about how the NFS client determines which UNIX-style credentials are used on the request. If those credentials are not consistent with what the NFS server is using, you may have problems with some operations such as file creation.

Usage Notes

1. The *directory_name* portion of the NFS path name is generally case sensitive. VM's minidisk file system and Shared File System are exceptions to this rule.

Mapping Macros

Mapping macros map the parameter options, constants, and data returned in many OpenExtensions callable services. Most of the mapping macros can be expanded with or without a DSECT statement. The invocation operand DSECT=YES is the default. The macros are described in Chapter 3, “Mapping Macro Descriptions,” on page 409.

Examples

The description of each callable service includes an invocation example. These examples follow the rules of reentrancy. They use DSECT=NO and place the variables in the program's dynamic storage DSECT, which is allocated upon entry. The declaration for all local variables used in an example follows the example.

Reentrant Coding versus Nonreentrant Coding: See “Example” on page 395 for an example of the w_getpsent (BPX1GPS) service using reentrant code. Compare this example with an example of nonreentrant code for the same service in “Nonreentrant Entry Linkage” on page 553, and note the following:

- Placement of the standard 18-word register save area
- Use of program/dynamic storage base registers
- @DYNAM DSECT in the reentrant version
- Different forms of the CALL macro
- Several variables (such as, PGPSCONTTYBLEN) that are initialized by the assembler in the nonreentrant version (see “BPXYPGPS — Map the Response Structure for the w_getpsent Service” on page 449 for the *DC*s), and at execution time with moves and stores in the reentrant version.

Callable Service Failures

When a typical application receives an unexpected return code from a callable service, it usually exits the application. If an application is written to handle or manage unexpected errors, you need to understand the following information.

Services can fail for a number of reasons: bugs in the system, user code causing failure return codes, or abend conditions. Depending on when the failure occurs in the service path, the requested function may or may not have been performed. For example, if the application provides an address for a file descriptor that does not exist, the open service (BPX1OPN) completes the open processing and then fails on the return path when trying to set the file descriptor. If an EFAULT return code is returned, the user may assume the file was not opened, even though it is.

If the return value parameter is not in valid storage, the services can complete successfully yet not return normally to the caller. Since the service cannot set the return value, it abends. It is possible for the C runtime library to convert the return value into a **SIGABND** or **SIGSEGV** signal that can be caught and handled by the user signal action defined in sigaction. The user needs to be aware that functions that abend in this way may have completed their processing. For example, a call to sigaction could modify the state of signal information and then fail on the return to the caller; in this case, the caller should not make any assumptions about the state of the signal environment.

Authorization

Users authorized to perform special functions are defined as having *appropriate privileges*, and they are called *superusers*. This corresponds to the user's process having an effective user ID of zero or the user's virtual machine having file pool administration authority for the applicable file pool server. For more information about POSIX user database concepts, see *z/VM: OpenExtensions User's Guide*.

Chapter 2. Callable Service Descriptions

This section describes each of the OpenExtensions callable services. The services are arranged in alphabetical order.

If you are unfamiliar with the conventions used to describe the system calls, refer to Chapter 1, “Invocation Details for Callable Services,” on page 1.

© Copyright IBM Corp. 1993, 2022

accept (BPX1ACP) — Accept a Connection Request from a Client Socket

BPX1ACP

socket_descriptor
sockaddr_length
sockaddr
return_value
return_code
reason_code

Purpose

Use the accept (BPX1ACP) service to allow a server to accept a connection request from a client. The service extracts the first connection on the queue of pending connections, creates a new socket with the same properties as the specified socket, and allocates a new descriptor for that socket. If there are no connections pending, the service either blocks until a connection request is received, or fails with an EWOULDBLOCK return code, depending on whether the specified socket is marked as blocking or nonblocking.

Parameters

socket_descriptor
(input,INT,4) is a variable for specifying the descriptor of the accepting (server) socket.

sockaddr_length
(input/output,INT,4) is a variable for specifying the length of the *sockaddr* parameter. The length should be less than 4096 bytes (4KB). On output, the service updates this field with the length of the client address returned in *sockaddr*. If you do not want the client address, specify `0` for this parameter.

sockaddr
(output,CHAR,*sockaddr_length*) is a variable where the service returns the SOCKADDR structure containing the socket address of the connecting client. The format of the socket address is determined by the domain in which the client resides. This field is mapped by the BPXYSOCK macro. See "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465.

return_value
(output,INT,4) is a variable where the service returns the new socket descriptor if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. The socket descriptor passed as input refers to a socket that was created with the socket (BPX1SOC) service, bound to an address with the bind (BPX1BND) service, and has successfully issued a call to the listen (BPX1LSN) service.

Before calling accept (BPX1ACP), you can find out if the socket has any connections pending by doing a read select with the select (BPX1SEL) service.

2. In order for a socket address to be returned for a UNIX domain socket, the client application doing the connect must bind a unique local name to the socket using the bind (BPX1BND) service before running the connect (BPX1CON) service.

Example

The following code accepts a connect request from a client. SOCKDESC was previously set by a call to socket (BPX1SOC). A bind (BPX1BND) and a listen (BPX1LSN) must also have been previously done. The SOCKADDR structure was built by the call to bind (BPX1BND). This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services” on page 465.

```
         CALL  BPX1ACP,              Accept a socket connect request   +
               (SOCKDESC,            Input: Socket descriptor          +
               =A(SOCK#LEN+SOCK_SUN#LEN), Input: Length - Sockaddr     +
               SOCKADDR,             Output: Sockaddr structure        +
               RETVAL,               Return value: Socket descr or -1  +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
         L  R2,RETVAL
         ST R2,SOCKDES2              Store the new socket descriptor
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	An incorrect file descriptor was specified. The following reason codes can accompany this return code: JRFileDesNotInUse, JRFileNotOpen.
ECMSPFSPERM	The physical file system encountered a system error. The following reason code can accompany this return code: JRInvalidVnode.
EINTR	A signal interrupted the accept service before any connections were available. The following reason code can accompany this return code: JRSignalReceived.
EINVAL	One of the input parameters was incorrect. The following reason codes can accompany this return code: JRNegativeValueInvalid, JRSocketCallParmError. The socket is not accepting connections. A listen must be done prior to the accept. The following reason code can accompany this return code: JRListenNotDone.
EIO	There has been a network or transport failure. The following reason code can accompany this return code: JRPrevSockError.
ENOBUFS	A buffer could not be obtained.
ENOTSOCK	*socket_descriptor* does not refer to a valid socket descriptor. The following reason code can accompany this return code: JRMustBeSocket.
EOPNOTSUPP	The socket type of the specified socket does not support accepting connections.
EWOULDBLOCK	The socket file descriptor is marked nonblocking, and no connections are present to be accepted.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “bind (BPX1BND) — Bind a Unique Local Name to a Socket Descriptor” on page 20
- “connect (BPX1CON) — Establish a Connection Between Two Sockets” on page 57
- “listen (BPX1LSN) — Prepare a Server Socket to Queue Incoming Connection Requests from Clients” on page 152
- “select/selectex (BPX1SEL) — Select on File Descriptors and Message Queues” on page 258
- “socket (BPX1SOC) — Create a Socket” on page 330

access (BPX1ACC) — Determine If a File Can Be Accessed

BPX1ACC

pathname_length
pathname
access_mode
return_value
return_code
reason_code

Purpose

Use the access (BPX1ACC) service to determine whether you can access a file. You identify the file by its path name.

Parameters

pathname_length
(input,INT,4) is a variable for specifying the length of the *pathname* parameter.

pathname
(input,CHAR,*pathname_length*) is a variable for specifying the path name of the file to be checked for accessibility. See "Understanding Byte File System (BFS) Path Name Syntax" on page 6.

access_mode
(input,INT,4) is a variable for specifying the accessibility to be tested. This variable is mapped by the BPXYACC macro. See "BPXYACC — Map Flag Values for the access Service" on page 412. The values for the variable are:

Value	Meaning
ACC_F_OK	Test for file existence.
ACC_R_OK	Test for permission to read.
ACC_W_OK	Test for permission to write.
ACC_X_OK	Test for permission to execute or search.

return_value
(output,INT,4) is a variable where the service returns `0` if the request completes successfully (the file exists or access is permitted), or `-1` if the request is not successful or the file cannot be accessed in the specified way.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. Testing for file permissions is based on the real user ID (UID) and real group ID (GID), not the effective UID or effective GID of the calling process.
2. The caller can test for the existence of a file, or for access to the file, but not both.

3. In testing for permission, the caller can test for any combination of read, write, and execute permission. If the caller is testing a combination of permissions, the return value indicates failure if any one of the accesses is not permitted.
4. If the caller has appropriate privileges, the access test is successful even if the permission bits are off, except when testing for execute permission. When the caller tests for execute permission, at least one of the execute permission bits must be on for the test to be successful.

Example

The following code determines if file **/usr/inv/network.t** can be accessed. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structure, see "BPXYACC — Map Flag Values for the access Service" on page 412.

```
         MVC   BUFFERA(18),=CL18'/usr/inv/network.t'
         MVC   BUFLENA,=F'18'
         XC    ACC(ACC#LENGTH),ACC
         MVI   ACCINTENTFLAGS,ACC_R_OK+ACC_W_OK  Read & write access
         SPACE ,
         CALL  BPX1ACC,              Determine accessibility of a file +
               (BUFLENA,             Input: Pathname length            +
               BUFFERA,              Input: Pathname                   +
               ACC,                  Input: Access, BPXYACC            +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
         SPACE ,
         ICM   R15,B'1111',RETVAL    Set condition code for RETVAL
         BZ    PSEUDO                Branch if RETVAL is zero
         CLC   RETCODE,=A(EACCES)    Compare RETCODE to EACCES
         BE    PSEUDO                Branch if access denied
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The calling process does not have appropriate permissions to access the file in the ways specified by the *access_mode* parameter, or the process does not have search permission for some component of the path name.
EINVAL	The *access_mode* parameter is incorrect. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRInvalidAmode.
ELOOP	A loop exists in symbolic links encountered during resolution of the *pathname* argument. This error is issued if more than 8 symbolic links are detected in the resolution of the path name.
ENAMETOOLONG	The path name is longer than 1023 characters, or some component of the path name is longer than 255 characters. CMS does not support name truncation.
ENOENT	No file named *pathname* was found, or no path name was specified. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRFileNotThere.
ENOTDIR	Some component of the path name is not a directory.

Return Code	Explanation
EROFS	The *access_mode* parameter is testing for write access to a read-only file system. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRReadOnlyFS.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “chmod (BPX1CHM) — Change the Mode of a File or Directory by Path Name” on page 28
- “open (BPX1OPN) — Open a File” on page 181
- “stat (BPX1STA) -- Get Status Information about a File by Path Name” on page 340.

alarm (BPX1ALR) — Set an Alarm

```
BPX1ALR
        seconds
        return_value
```

Purpose

Use the alarm (BPX1ALR) service to generate a SIGALRM signal after a specified number of seconds have elapsed. The SIGALRM signal delivery is directed to the calling thread.

Parameters

seconds
(input,INT,4) is a variable for specifying an unsigned value which is the minimum number of seconds to pass between receipt of this request and generation of the SIGALRM signal. If zero is specified, any outstanding alarm request is canceled; no new alarm interval is set. Processor scheduling delays can cause the delivery of the SIGALRM signal to occur after the desired time.

return_value
(output,INT,4) is a variable where the service stores an unsigned return value. If there is a previous alarm request with time remaining, the service returns a nonzero value that is the number of seconds until the previous request would have generated a SIGALRM signal. The return value is rounded to the nearest second except when the time remaining is less than a half second. When the remaining time is less than a half second and greater than zero, the return value is set to 1. If there is no previous alarm request with time remaining, the return value is set to 0.

Usage Notes

1. The access (BPX1ACC) service is always successful, and no return value is reserved to indicate an error.
2. An abend is generated when failures are encountered that prevent the access (BPX1ACC) service from completing successfully.
3. Alarm requests are not stacked; only one SIGALRM generation is scheduled in this manner. If SIGALRM was not generated, the call reschedules the time that SIGALRM is generated.

Characteristics and Restrictions

See Appendix E, “The Relationship of OpenExtensions Signals to Callable Services,” on page 557.

Example

The following code schedules an alarm in 5 seconds. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   SECONDS,=F'5'
         SPACE ,
         CALL  BPX1ALR,              Schedule Alarm                    +
               (SECONDS,             Input: Time before SIGALRM        +
               RETVAL),              Return value: 0 or -1             +
               VL,MF=(E,PLIST)       ----------------------------------
```

VM-Related Information

Both the alarm service, BPX1ALR, and the sleep service, BPX1SLP, use CMS Application Multitasking Timer Services. If the task invokes TimerStopAll, any outstanding timers set by the alarm or sleep service will also be canceled.

If a timer set by the alarm or sleep service is canceled (using TimerStopAll) or expires, both a SIGALRM signal is generated and a VMTIMER event is signalled. See *z/VM: CMS Application Multitasking* for more information on TimerStopAll and the VMTIMER event.

Related Services

Other callable services related to this service are:

- “exec (BPX1EXC) — Run a Program” on page 72
- “sigaction (BPX1SIA) — Examine or Change a Signal Action” on page 315
- “sigprocmask (BPX1SPM) — Examine or Change a Thread's Signal Mask” on page 321
- “sleep (BPX1SLP) — Suspend Execution of a Process for an Interval of Time” on page 328
- “spawn (BPX1SPN) — Spawn a Process” on page 333.

bind (BPX1BND) — Bind a Unique Local Name to a Socket Descriptor

BPX1BND

socket_descriptor
sockaddr_length
sockaddr
return_value
return_code
reason_code

Purpose

Use the bind (BPX1BND) service to bind a unique local name to a socket descriptor.

Parameters

socket_descriptor
(input,INT,4) is a variable for specifying the descriptor of the socket to be bound.

sockaddr_length
(input,INT,4) is a variable for specifying the length of the *sockaddr* parameter.

sockaddr
(input,CHAR,*sockaddr_length*) is a variable for specifying the SOCKADDR structure that contains the name to be bound to the socket descriptor. The format of SOCKADDR is determined by the domain in which the socket descriptor was created. SOCKADDR is mapped by the BPXYSOCK macro. See "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. An application can retrieve the assigned socket name with the getsockname service.
2. Sockets in the AF_UNIX domain create a name in the file system that must be deleted by the application (using unlink) when it is no longer needed.
3. For SOCKADDR to be returned on an accept request for an AF_UNIX domain socket, the client application doing the connect must bind a unique local SOCKADDR to the socket with the bind request before issuing the connect request.
4. Server applications issue the bind request to register their addresses with the system. Both connection and connectionless servers must do this before accepting requests from clients.
5. For AF_INET or AF_INET6, the user must have appropriate privileges to bind to a port in the range from 1 to 1023.

6. For AF_IUCV, the local socket name must be unique within the virtual machine. Only one socket can be bound to a given name. The recommended form of the name contains eight characters, padded with blanks to the right. The eight characters for a connect call executed by a client must exactly match the eight characters passed in the bind call executed by the server.

Example

The following code does a bind to associate a name with a socket. SOCKDESC was previously set by a call to socket (BPX1SOC). This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services” on page 465.

```
         SPACE ,
         MVI   SOCK_LEN,12            Store the length of the address
         MVI   SOCK_FAMILY,AF_UNIX    Set the domain to AF_UNIX
         MVC   SOCK_SUN_NAME(12),=CL12'/tmp/socket1'  Set the name
         CALL  BPX1BND,               Bind a name to a socket           +
               (SOCKDESC,             Input: Socket Descriptor          +
               SOCK#LEN+SOCK_SUN#LEN, Input: Length - Sockaddr         +
               SOCKADDR,              Input: Sockaddr structure         +
               RETVAL,                Return value: 0 or -1             +
               RETCODE,               Return code                       +
               RSNCODE),              Reason code                       +
               VL,MF=(E,PLIST)        ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	For AF_UNIX, the process does not have search permission on a component of the path prefix, or it does not have write access to the directory of the requested name. For AF_INET or AF_INET6, permission denied. A user that is not in the obeylist attempts to bind to a port between 0-1023 when RESTRICTLOWPORTS has been specified on the ASSORTEDPARMS statement. Or, a user attempts to bind to a port that has previously been reserved using a PORT statement in the TCP/IP configuration file or through obeyfile processing.
EADDRINUSE	For AF_INET, AF_INET6, and AF_IUCV, the specified address is already in use.
EAFNOSUPPORT	The address family specified in the address structure is not supported.
EBADF	The socket descriptor is incorrect. The following reason codes can accompany this return code: JRFileDesNotInUse, JRFileNotOpen.
EDESTADDRREQ	A destination address is required. The following reason code can accompany this return code: JRSocketCallParmError.
EINVAL	One of the input parameters is incorrect. The following reason codes can accompany this return code: JRSocketCallParmError, JRSockNoname.For AF_UNIX, the following reason codes can accompany this return code: JREndingSlashExtLink, JRNFSNotallowed, JRInvalidExtLinkLen
EIO	There has been a network or transport failure. The following reason code can accompany this return code: JRPrevSockError.
ENOBUFS	A buffer could not be obtained.
ENOTSOCK	*socket_descriptor* does not refer to a valid socket descriptor. The following reason code can accompany this return code: JRMustBeSocket.
EPERM	The user is not permitted to bind to the specified port. The following reason code can accompany this return code: JRUserNotPrivileged.

The following are for AF_UNIX only:

Return Code	Explanation
EEXIST	The file or socket already exists. The following reason code can accompany this return code: JRExtFileAlreadyExists
EIO	An I/O error occurred.
ELOOP	Too many symbolic links were encountered in translating the path name in *sockaddr*.
ENAMETOOLONG	A component of a path name exceeded NAME_MAX characters, or an entire path name exceeded PATH_MAX characters.
ENOENT	The AF_UNIX path name is not valid. The following reason code can accompany this return code: JRFileNotThere
ENOTDIR	A component of the path prefix of the path name in *sockaddr* is not a directory.
EROFS	The name would reside on a read-only file system. The following reason code can accompany this return code: JRReadOnlyFS

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “accept (BPX1ACP) — Accept a Connection Request from a Client Socket” on page 12
- “connect (BPX1CON) — Establish a Connection Between Two Sockets” on page 57
- “getsockname/getpeername (BPX1GNM) — Get the Name of a Socket or Peer” on page 136
- “listen (BPX1LSN) — Prepare a Server Socket to Queue Incoming Connection Requests from Clients” on page 152
- “socket (BPX1SOC) — Create a Socket” on page 330

chaudit (BPX1CHA) — Change Audit Flags for a File by Path Name

```
BPX1CHA
        pathname_length
        pathname
        audit_flags
        option_code
        return_value
        return_code
        reason_code
```

Purpose

Use the chaudit (BPX1CHA) service to change the types of access to a file to be audited for the external security manager (ESM). You identify the file by its path name.

For the corresponding service using a file descriptor, see "fchaudit (BPX1FCA) — Change Audit Flags for a File by Descriptor" on page 81.

Parameters

pathname_length
(input,INT,4) is a variable for specifying the length of the *pathname* parameter.

pathname
(input,CHAR,*pathname_length*) is a variable for specifying the path name of the file for which auditing is to be changed. See "Understanding Byte File System (BFS) Path Name Syntax" on page 6.

audit_flags
(input,INT,4) is a variable for specifying the access to be audited. This variable is mapped by the BPXYAUDT macro. See "BPXYAUDT — Map Flag Values for the chaudit and fchaudit Services" on page 413. Valid values for this variable include any combination of the following:

Value	Meaning
AUDTREADFAIL	Audit failing read requests.
AUDTREADSUCCESS	Audit successful read requests.
AUDTWRITEFAIL	Audit failing write requests.
AUDTWRITESUCCESS	Audit successful write requests.
AUDTEXECFAIL	Audit failing execute or search requests.
AUDTEXECSUCCESS	Audit successful execute or search requests.

option_code
(input,INT,4) is a variable for specifying whether you are changing the auditing for the user or for the security auditor. This variable can have the following values:

0
The user's auditing is being changed.

1
The security auditor's auditing is being changed.

return_value
(output,INT,4) is a variable where the service returns 0 if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. Audit flags are stored with every object in the Byte File System. They are intended for use by an External Security Manager (ESM) and are not used by native BFS server security or auditing functions. You can use the chaudit (BPX1CHA) service to change any of the audit flags, even when there is no ESM installed. However, because native BFS does not use the audit flags, they have no effect on security or auditing if no ESM is installed.
2. When no ESM is installed, the authority required to use this service is defined as follows:
   - To change the user audit flags, the user must be either a superuser or the owner of the file.
   - To change the auditor audit flags, the user must be a superuser.
3. When an ESM is installed, the authority requirements to use this service are defined by the ESM. For example, the ESM could define a level of authority called auditor authority, and further declare that auditor authority is required to change the auditor audit flags.

Example

The following code changes the audit flags for the file identified by path name. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYAUDT — Map Flag Values for the chaudit and fchaudit Services” on page 413.

```
         MVC   BUFFERA(18),=CL18'/usr/inv/network.t'
         MVC   BUFLENA,=F'18'
         MVI   AUDTREADACCESS,AUDTREADFAIL
         MVI   AUDTWRITEACCESS,AUDTWRITEFAIL
         MVI   AUDTEXECACCESS,AUDTEXECFAIL
         MVI   AUDTRSRV,0
         SPACE ,
         CALL  BPX1CHA,              Change audit                      +
               (BUFLENA,             Input: Pathname length            +
               BUFFERA,              Input: Pathname                   +
               AUDT,                 Input: Audit flags, BPXYAUDT      +
               =F'0',                Input: 0 user, 1 security auditor +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The calling process does not have search permission for some component of the path name.

Return Code	Explanation
EINVAL	The *option_code* parameter is incorrect. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRBadAuditOption
ELOOP	A loop exists in symbolic links encountered during resolution of the *pathname* argument. This error is issued if more than 8 symbolic links are detected in the resolution of the path name.
ENAMETOOLONG	The path name is longer than 1023 characters, or a component of the path name is longer than 255 characters. CMS does not support name truncation.
ENOENT	No file named *pathname* was found, or no path name was specified. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRFileNotThere.
ENOTDIR	A component of the path name is not a directory.
EPERM	The effective UID of the calling process does not match the file's owner UID; or the calling process does not have appropriate privileges; or, if *option_code* indicated that the auditor audit flags were to be changed, the user does not have auditor authority.
EROFS	The file exists on a read-only file system. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRReadOnlyFS.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “fchaudit (BPX1FCA) — Change Audit Flags for a File by Descriptor” on page 81
- “stat (BPX1STA) -- Get Status Information about a File by Path Name” on page 340.

chdir (BPX1CHD) — Change the Working Directory

BPX1CHD

pathname_length
pathname
return_value
return_code
reason_code

Purpose

Use the chdir (BPX1CHD) service to change your working directory from the current one to a new one. The working directory is the starting point for path searches of path names not beginning with a slash. You identify the new directory by its path name.

Parameters

pathname_length
(input,INT,4) is a variable for specifying the length of the *pathname* parameter.

pathname
(input,CHAR,*pathname_length*) is a variable for specifying the path name of the directory you want to become your new working directory. See "Understanding Byte File System (BFS) Path Name Syntax" on page 6.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Example

The following code changes the working directory for the task. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551.

```
         MVC   BUFFERA(8),=CL8'/usr/inv'
         MVC   BUFLENA,=F'8'
         SPACE ,
         CALL  BPX1CHD,              Change working directory          +
               (BUFLENA,             Input: Pathname length            +
               BUFFERA,              Input: Pathname                   +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The calling process does not have permission to search one of the components of the path name.
EINVAL	The *pathname* parameter is not valid; it contains nulls.
ELOOP	A loop exists in symbolic links encountered during resolution of the *pathname* argument. This error is issued if more than 8 symbolic links are detected in the resolution of the path name.
ENAMETOOLONG	The path name is longer than 1023 characters, or a component of the path name is longer than 255 characters. CMS does not support name truncation.
ENOENT	No directory named *pathname* was found, or no path name was specified. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRChdNoEnt and JRQuiescing.
ENOTDIR	Some component of the path name is not a directory. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRChdNotDir.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “closedir (BPX1CLD) — Close a Directory” on page 36
- “getcwd (BPX1GCW) — Get the Path Name of the Working Directory” on page 112
- “mkdir (BPX1MKD) — Make a Directory” on page 160
- “opendir (BPX1OPD) — Open a Directory” on page 185
- “readdir (BPX1RDD) — Read an Entry from a Directory” on page 231
- “rmdir (BPX1RMD) — Remove a Directory” on page 256
- “unlink (BPX1UNL) — Remove a Directory Entry” on page 379.

chmod (BPX1CHM) — Change the Mode of a File or Directory by Path Name

BPX1CHM

pathname_length
pathname
mode
return_value
return_code
reason_code

Purpose

Use the chmod (BPX1CHM) service to modify the permission bits that control the owner access, group access, and general access to the file. You can use this service to set flags that modify the user ID (UID) and group ID (GID) of the file when it is executed. You can also use this service to set the sticky bit to indicate from where the file should be fetched. You identify the file by its path name.

For the corresponding service using a file descriptor, see "fchmod (BPX1FCM) — Change the Mode of a File or Directory by Descriptor" on page 84.

Parameters

pathname_length
(input,INT,4) is a variable for specifying the length of the *pathname* parameter.

pathname
(input,CHAR,*pathname_length*) is a variable for specifying the path name of the file whose mode you want to change. See "Understanding Byte File System (BFS) Path Name Syntax" on page 6.

mode
(input,INT,4) is a variable for specifying the new mode of the file. This parameter, which is mapped by the BPXYMODE macro, specifies the file type and the permissions you grant to yourself, to your group, and to any user. See "BPXYMODE — Map Mode Constants" on page 437 for the parameter options.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. File descriptors that are open when the chmod (BPX1CHM) service is called retain the access permission they had when the file was opened.
2. For mode bits to be changed, the effective UID of the calling process must match the file's owner UID, or the caller must have appropriate privileges.
3. When the mode is changed successfully, the file's change time is updated as well.

4. Setting the set-group-ID-on-execution permission means that when this file is run, through the exec service, the effective GID of the caller is set to the file's owner GID, so that the caller seems to be running under the GID of the file, rather than that of the actual invoker.

   The set-group-ID-on-execution permission is set to zero if both of the following are true:

   - The caller does not have appropriate privileges.
   - The GID of the file's owner does not match the effective GID or one of the supplementary GIDs of the caller.
5. Setting the set-user-ID-on-execution permission means that when this file is run, the process's effective UID is set to the file's owner UID, so that the process seems to be running under the UID of the file's owner, rather than that of the actual invoker.

Example

The following code changes the file mode for the file identified by path name. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYMODE — Map Mode Constants” on page 437.

```
         MVC   BUFFERA(26),=CL26'newprogs/path/eightfold.c'
         MVC   BUFLENA,=F'26'
         XC    S_MODE,S_MODE
         MVI   S_MODE2,S_IRUSR       All read and write
         MVI   S_MODE3,S_IWUSR+S_IRGRP+S_IWGRP+S_IROTH+S_IWOTH
         SPACE ,
         CALL  BPX1CHM,              Change File Modes                +
               (BUFLENA,             Input: Pathname length           +
               BUFFERA,              Input: Pathname                  +
               S_MODE,               Input: Mode, mapped by BPXYMODE  +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The calling process does not have permission to search some component of the path name.
ELOOP	A loop exists in symbolic links encountered during resolution of the *pathname* argument. This error is issued if more than 8 symbolic links are detected in the resolution of the path name.
ENAMETOOLONG	The path name is longer than 1023 characters, or a component of the path name is longer than 255 characters. CMS does not support filename truncation.
ENODEV	An attempt was made to use a character special file for a device not supported by OpenExtensions.
ENOENT	No file named *pathname* was found, or no path name was specified. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRFileNotThere.
ENOTDIR	Some component of the path name is not a directory.
EPERM	The effective UID of the calling process does not match the owner of the file, and the calling process does not have appropriate privileges.

Return Code	Explanation
EROFS	The *pathname* parameter specifies a file that is on a read-only file system. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRReadOnlyFS.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “chown (BPX1CHO) — Change the Owner or Group of a File or Directory” on page 31
- “fchmod (BPX1FCM) — Change the Mode of a File or Directory by Descriptor” on page 84
- “mkdir (BPX1MKD) — Make a Directory” on page 160
- “open (BPX1OPN) — Open a File” on page 181
- “stat (BPX1STA) -- Get Status Information about a File by Path Name” on page 340.

chown (BPX1CHO) — Change the Owner or Group of a File or Directory

BPX1CHO

pathname_length
pathname
owner_UID
group_ID
return_value
return_code
reason_code

Purpose

Use the chown (BPX1CHO) service to change a file's owner, group, or both. You identify the file by its path name.

For the corresponding service using a file descriptor, see "fchown (BPX1FCO) — Change the Owner and Group of a File or Directory by Descriptor" on page 86.

Parameters

pathname_length
(input,INT,4) is a variable for specifying the length of the *pathname* parameter.

pathname
(input,CHAR,*pathname_length*) is a variable for specifying the path name of the file for which you wish to change the owner or group or both. See "Understanding Byte File System (BFS) Path Name Syntax" on page 6.

owner_UID
(input,INT,4) is a variable for specifying the new owner UID assigned to the file, or the present value if there is no change. This parameter must be specified.

group_ID
(input,INT,4) is a variable for specifying the new group ID assigned to the file, or the present value if there is no change. This parameter must be specified.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. The chown (BPX1CHO) service changes the owner UID and owner GID of a file. Only a superuser can change the owner UID of a file.
2. The owner GID of a file can be changed by a caller if the caller has appropriate privileges, or if a caller meets all of these conditions:

- The effective UID of the caller matches the file's owner UID.
- The *owner_UID* value specified in the change request matches the file's owner UID.
- The *group_ID* value specified in the change request is the effective GID, or one of the supplementary GIDs, of the caller.

3. The set-user-ID-on-execution and set-group-ID-on-execution permissions of the file mode are automatically turned off.
4. If the change request is successful, the change time for the file is updated.
5. Values for both *owner_UID* and *group_ID* must be specified as they are to be set. If it is desired to change only one of these values, the other must be set to its present value to remain unchanged.

Example

The following code changes the owner of **/somedir/somefile.c** from the current owner to that specified by USERID and GROUPID. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   BUFFERA(20),=CL20'/somedir/somefile.c'
         MVC   BUFLENA,=F'20'
         MVC   USERID,..             New owner UID from stat
         MVC   GROUPID,..            New owner GID from stat
         SPACE ,
         CALL  BPX1CHO,              Change owner and group of a file  +
               (BUFLENA,             Input: Pathname length            +
               BUFFERA,              Input: Pathname                   +
               USERID,               Input: New owner UID              +
               GROUPID,              Input: New owner GID              +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The calling process does not have permission to search some component of the path name.
EINVAL	The *owner_UID* or *group_ID* parameter is incorrect
ELOOP	A loop exists in symbolic links encountered during resolution of the *pathname* argument. This error is issued if more than 8 symbolic links are detected in the resolution of the path name.
ENAMETOOLONG	The path name is longer than 1023 characters, or a component of the path name is longer than 255 characters. CMS does not support name truncation.
ENODEV	An attempt was made to use a character special file for a device not supported by OpenExtensions.
ENOENT	No file named *pathname* was found, or no path name was specified. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRFileNotThere.
ENOTDIR	Some component of the path name is not a directory.
EPERM	The calling process does not have appropriate privileges.
EROFS	The *pathname* parameter specifies a file on a read-only file system. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRReadOnlyFS.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “fchown (BPX1FCO) — Change the Owner and Group of a File or Directory by Descriptor” on page 86
- “fstat (BPX1FST) -- Get Status Information about a File by Descriptor” on page 102
- “lstat (BPX1LST) — Get Status Information about a File or Symbolic Link by Path Name” on page 157
- “stat (BPX1STA) -- Get Status Information about a File by Path Name” on page 340.

close (BPX1CLO) — Close a File or Socket

BPX1CLO

file_descriptor
return_value
return_code
reason_code

Purpose

Use the close (BPX1CLO) service to close a file or socket.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the descriptor of the file or socket you want to close.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. Closing a file closes, or frees, the file descriptor by which the file was known to the process. The system can then reassign the file descriptor to the same file or to another file when it is opened.
2. Closing a file descriptor also unlocks all outstanding byte range locks that a process has on the associated file.
3. If a file has been opened by more than one process, each process has a file descriptor. When the last open file descriptor is closed, the file itself is closed. If the file's link count is zero at that time, the file's space is freed and the file becomes inaccessible. When the last open file descriptor for a pipe or FIFO special file is closed, any data remaining in the file is discarded.

Example

The following code closes the standard input file. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         CALL  BPX1CLO,              Close a file                    +
               (=A(STDIN_FILENO),    Input: File descriptor          +
               RETVAL,               Return value: 0 or -1           +
               RETCODE,              Return code                     +
               RSNCODE),             Reason code                     +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EAGAIN	The service did not complete, because the file descriptor specified is currently in use by another thread in the same process.
EBADF	The *file_descriptor* parameter does not identify a valid, open file. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRClNeedClose and JRNotForDir.
EINTR	The service was interrupted by a signal while it was processing the close request. The file may or may not be closed.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “exec (BPX1EXC) — Run a Program” on page 72
- “fcntl (BPX1FCT) — Control Open File Descriptors” on page 88
- “fork (BPX1FRK) — Create a New Process” on page 96
- “open (BPX1OPN) — Open a File” on page 181
- “pipe (BPX1PIP) — Create an Unnamed Pipe” on page 199
- “socket (BPX1SOC) — Create a Socket” on page 330
- “spawn (BPX1SPN) — Spawn a Process” on page 333
- “unlink (BPX1UNL) — Remove a Directory Entry” on page 379.

closedir (BPX1CLD) — Close a Directory

BPX1CLD

directory_file_descriptor

return_value

return_code

reason_code

Purpose

Use the closedir (BPX1CLD) service to close a directory. You identify the directory by its directory file descriptor.

Parameters

directory_file_descriptor
(input,INT,4) is a variable for specifying the directory file descriptor of the directory you want to close. This value was returned when the directory was opened.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Example

The following code closes the directory identified by FILEDESC. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   FILEDESC,..          Directory descriptor from opendir
         SPACE ,
         CALL  BPX1CLD,             Close a directory                +
               (FILEDESC,           Input: Directory file descriptor +
               RETVAL,              Return value: 0 or -1            +
               RETCODE,             Return code                      +
               RSNCODE),            Reason code                      +
               VL,MF=(E,PLIST)      ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	The *directory_file_descriptor* parameter does not represent an open directory.
EINTR	The service was interrupted by a signal while it was processing the request. The directory may or may not be closed.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “opendir (BPX1OPD) — Open a Directory” on page 185
- “readdir (BPX1RDD) — Read an Entry from a Directory” on page 231
- “rewinddir (BPX1RWD) — Reposition a Directory Stream to the Beginning” on page 254.

cmsprocclp (BPX1MPC) — Clean Up Kernel Resources

BPX1MPC

status_field
return_value
return_code
reason_code

Purpose

Use the cmsprocclp (BPX1MPC) service to clean up the OpenExtensions-related resources for an entire process or on a thread-by-thread basis.

Parameters

status_field
(input,INT,4) is a variable for specifying exit status values. If the invocation of this service causes a full process cleanup to occur, and the contents of the status field conform to the allowable exit status values, the contents are made available to the parent when the wait service is issued. For the mapping of this parameter and a description of the allowable exit status values, see "BPXYWAST — Map the Wait Status Word" on page 486.

return_value
(output,INT,4) is a variable where the service returns one of the following values:

Value	Explanation
0	OpenExtensions thread-related resources were cleaned up for the calling thread.
1	OpenExtensions process-related resources were cleaned up for the calling process.
-1	The service failed to clean up process resources.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Note

The cmsprocclp (BPX1MPC) service normally cleans up just the thread-related data for the calling thread. The following two situations, however, cause full process cleanup to occur:

- If the call is made from the initial thread of the process and no other threads exist in the process.
- If the call is made from the last thread that is left in the process, and that thread is not the initial thread, and the initial thread has not performed any OpenExtensions system calls.

In these two cases, both the OpenExtensions thread-related and process-related resources are cleaned up and OpenExtensions process termination is performed. See the _exit (BPX1EXI) service for a description of ending an OpenExtensions process.

Example

The following code causes all OpenExtensions related resources to be released for this thread, and if this is the last OpenExtensions thread in the process, for the process. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structure, see "BPXYWAST — Map the Wait Status Word" on page 486.

```
         XC    WAST(WAST#LENGTH),WAST
         MVI   WASTEXITCODE,57       User defined exit code
         SPACE ,
         CALL  BPX1MPC,              CMS Process cleanup               +
               (WAST,                Input: Ending status code 0-255   +
               RETVAL,               Return value: 0, -1 or 1          +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return code:

Return Code	Explanation
ECMSERR	The call was unsuccessful due to a CMS environmental or internal error. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRInvTermStat.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, "Return Codes," on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, "Reason Codes," on page 495.

Related Services

Other callable services related to this service are:

- "_exit (BPX1EXI) — End a Process and Bypass the Cleanup" on page 79
- "wait (BPX1WAT) — Wait for a Child Process to End" on page 385.

cmssigsetup (BPX1MSS) — Set Up CMS Signals

BPX1MSS

signal_interface_routine_address
user_data
default_override_signal_set
default_terminate_signal_set
return_value
return_code
reason_code

Purpose

Use the cmssigsetup (BPX1MSS) service to catch or intercept signals. This service also allows you to intercept cancellation and quiesce interrupts. Only one cmssigsetup (BPX1MSS) service can be active in a process. If you must perform a second cmssigsetup (BPX1MSS) service in a process, you must first use the cmsunsigsetup (BPX1MSD) service on the thread that issued the cmssigsetup (BPX1MSS) service request before you call the cmssigsetup (BPX1MSS) service again. Both CMS thread termination and the cmsprocclp (BPX1MPC) service perform the cmsunsigsetup (BPX1MSD) service.

Parameters

signal_interface_routine_address
(input,PTR,4) is a variable for specifying the address of the user-supplied signal interface routine (SIR) that gets control when a signal handler needs to be invoked. The signal handler is defined by the sigaction (BPX1SIA) call. You can also invoke the SIR to process a default signal action, depending on the values specified for *default_override_signal_set*.

user_data
(input,CHAR,4) is a variable for specifying 4 bytes of user-supplied data to be passed to the signal interrupt routine on invocation from signal processing.

default_override_signal_set
(input,CHAR,8) is a variable for specifying a 64-bit mask of signals that the SIR processes when their respective default actions take place. The leftmost bit represents signal number 1 and the rightmost bit represents signal number 64. The signals SIGKILL and SIGSTOP cannot be intercepted. The bit positions that represent these signals are ignored. Signal 64 represents cancellation or quiesce requests.

default_terminate_signal_set
(input,CHAR,8) is a variable for specifying a 64-bit mask of signals specified in the *default_override_signal_set* parameter that also causes the process to end. The leftmost bit represents signal number 1 and the rightmost bit represents signal number 64. When a signal bit is set to 1, the signal that it represents interrupts a task that is either stopped or in a wait state. It is up to the signal interrupt routine to end the process. The bit that represents signal 64 of this mask is reserved.

return_value
(output,INT,4) is a variable where the service returns 0 if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. A process image that results after the exec (BPX1EXC) service is not set up for signals.
2. The signal delivery data structure is mapped by the BPXYPPSD macro. See "BPXYPPSD — Map the Signal Delivery Data Structure" on page 451.
3. The SIR receives control with the following register interface:

Register	Contents
Reg 0	0
Reg 1	Address of standard parameter list. PARM1= address of BPXYPPSD; Reg 1 = ADDR(PpsdSirPARMS).
Regs 2–12	0
Reg 13	0 No save area for registers is provided to the SIR. The SIR does not save caller's registers.
Reg 14	0 No return address.
Reg 15	Set to address of the SIR.

4. The SIR receives control in the following system states:

   Amode:
   31-bit

   ASC mode:
   Primary mode

   Interrupt status:
   Enabled for interrupts

   Signal Mask:
   All signals that may be blocked by the signal mask are blocked.

5. Following are the steps that a user-supplied SIR must perform.

   a. The SIR must obtain local storage for a local copy of the BPXYPPSD and copy the BPXYPPSD information into this local storage.

   b. The PPSD contains the information necessary for the SIR to determine the reason for the interruption. The interruption can be the result of a signal, cancellation, or quiesce request.

   c. If the interrupt cannot be processed at this time, possibly due to general register 13 not currently containing the address of a program stack, or the last service called on the current thread was cond_setup, then the queue_interrupt (BPX1SPB) service request is issued. (See "queue_interrupt (BPX1SPB) — Return the Last Interrupt Delivered" on page 223.) Then go to step "5.h" on page 42.

   d. If the interrupt is a signal and the default action is to be performed by the SIR, write the appropriate messages to the terminal and end the process. For more information on how to end the process, see "_exit (BPX1EXI) — End a Process and Bypass the Cleanup" on page 79.

   e. If the interrupt is a cancellation or a terminating quiesce request, cleanup any necessary thread related resources and end the thread. To end the thread issue the pthread_exit service with *options_field* set to PTEXITTHREAD. If the interrupt is because of a cancellation, issue the pthread_exit service with *status_field* set to -1. For more information on how to end the thread, see "pthread_exit_and_get (BPX1PTX) — Exit and Get a New Thread" on page 209. The SIR will receive these types of interrupts only if bit 64 of *default_override_signal_set* is set on.

   f. Obtain language stack storage for the signal handler.

g. Set the signal processor mask to the appropriate value before invoking the signal handler. This mask is formed by taking the union of the current signal mask and the value of *sa_mask* specified on the sigaction call for the signal being delivered, and then including the signal being delivered. The signal processor mask is set by calling the sigprocmask service. Recursive calls to the SIR can occur after calling the BPX1SPM service here to unblock signals. Therefore, the SIR cannot use the BPXYPPSD macro after calling the BPX1SPM service.

h. Conform to the language-dependent requirements for invoking signal-handlers.

i. On return from the signal handler, call the BPX1SPM service to set the signal processor mask to the interrupted value that was saved in the BPXYPPSD macro on entry to this SIR.

j. Use the CSRL16J CMS service to load 16 registers and jump to the address that was interrupted by the signal.

6. The use of the *default_terminate_signal_set* is to indicate to the OpenExtensions kernel which signals intercepted by the SIR cause the process to end. An example of usage might be that a user wishes to intercept the SIGUSR1 signal, but instead of performing the OpenExtensions default of termination, it wishes to issue a message and then throw the signal away (ignore it). In this case, the user would turn the corresponding bit on in the *default_override_signal_set* and off in the *default_terminate_signal_set*. This bit set combination tells the kernel not to interrupt functions that return an EINTR.

Characteristics and Restrictions

See Appendix E, “The Relationship of OpenExtensions Signals to Callable Services,” on page 557.

Example

The following code allows the invoker to catch signals. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
* Each bit of the mask represents a signal 1-64.
         MVC   INTMASK(8),=XL8'F000000000000000'    Default sig 1-4
         MVC   TERMMASK(8),=XL8'F000000000000000'   Terminate sig 1-4
         LA    R15,BUFFERA
         ST    R15,USERWORD
         SPACE ,
         CALL  BPX1MSS,              Register CMS signals, this task   +
               (=V(SIRTN),           Input: Signal interrupt routine   +
               USERWORD,             Input: User data                  +
               INTMASK,              Input: Default override signals   +
               TERMMASK,             Input: Default terminate signals  +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return code:

Return Code	Explanation
ECMSINITIAL	The service failed. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRAlreadySigSetup.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “alarm (BPX1ALR) — Set an Alarm” on page 18
- “exec (BPX1EXC) — Run a Program” on page 72
- “kill (BPX1KIL) — Send a Signal to a Process” on page 146
- “pthread_cancel (BPX1PTB) — Cancel a Thread” on page 201
- “sigaction (BPX1SIA) — Examine or Change a Signal Action” on page 315
- “sigprocmask (BPX1SPM) — Examine or Change a Thread's Signal Mask” on page 321
- “sigsuspend (BPX1SSU) — Change the Signal Mask and Suspend the Thread Until a Signal Is Delivered” on page 324.

cmsunsigsetup (BPX1MSD) — Detach the Signal Setup

BPX1MSD

signal_interface_routine_address
user_data
default_override_signal_set
default_terminate_signal_set
return_value
return_code
reason_code

Purpose

Use the cmsunsigsetup (BPX1MSD) service to delete the signal setup established by the cmssigsetup (BPX1MSS) service. The parameters specified in the cmssigsetup (BPX1MSS) service are returned by the cmsunsigsetup (BPX1MSD) service. The signal actions for all signals in the process set by the sigaction (BPX1SIA) service are set to default action SIG_DFL.

Parameters

signal_interface_routine_address
(output,PTR,4) is a variable where the *signal_interface_routine_address* set by the cmssigsetup (BPX1MSS) service is returned.

user_data
(output,INT,4) is a variable where the *user_data* set by the cmssigsetup (BPX1MSS) service is returned.

default_override_signal_set
(output,CHAR,8) is a variable where the *default_override_signal_set* set by the cmssigsetup (BPX1MSS) service is returned.

default_terminate_signal_set
(output,CHAR,8) is a variable where the *default_terminate_signal_set* set by the cmssigsetup (BPX1MSS) service is returned.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Characteristics and Restrictions

See Appendix E, “The Relationship of OpenExtensions Signals to Callable Services,” on page 557.

Example

The following code detachs the invoker from being able to catch signals. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         CALL  BPX1MSD,              Reregister CMS signals, this task +
               (SIRTNA,              Signal interface routine address  +
               USERWORD              User data
               INTMASK,              Default override signal set       +
               TERMMASK,             Default terminate signal set      +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return code:

Return Code	Explanation
ECMSINITIAL	The service failed. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRNotSigSetup.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Another callable service related to this service is:

- “cmssigsetup (BPX1MSS) — Set Up CMS Signals” on page 40

cond_cancel (BPX1CCA) — Cancel Interest in Events

BPX1CCA

return_value

return_code

reason_code

Purpose

Use the cond_cancel (BPX1CCA) service to cancel the interest in event notifications. This call cancels the effects of a previous call to the cond_setup (BPX1CSE) service.

Parameters

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. The intended use of cond_cancel is for a program to clean up when it has used the cond_setup service, but does not call cond_wait or cond_timed_wait. The cond_setup service causes the thread to be eligible to receive event notifications. If the program running on the thread is no longer interested in these events, it should call cond_cancel to tell the system that event notifications are no longer required.
2. If you intend at some later time to call cond_wait or cond_timed_wait to wait until some event occurs, use the cond_setup service to make your program eligible to receive event notifications. The system notes that your program will be waiting for some other thread to either send it a signal or else to use the cond_post service to send an event notification. Both of these require use of CMS services. If CMS determines that it has become impossible to send a signal or event notification to your program, it checks whether your program is or will be calling either of the cond_wait or cond_timed_wait services. If so, CMS abnormally terminates your program to prevent it from waiting for something that cannot occur. For this reason, if your program uses the cond_setup service but does not subsequently call either cond_wait or cond_timed_wait, it should use the cond_cancel service to cancel the setup to receive event notifications.
3. When the program cannot determine whether cond_wait or cond_timed_wait has been called, it should call cond_cancel to ensure that the thread is not eligible to receive event notifications.

Example

The following code demonstrates how to cancel a program's interest in events that were selected by a call to the cond_setup service. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         CALL  BPX1CCA,             Cancel cond_setup               +
               (RETVAL,             Return value: 0 or -1           +
               RETCODE,             Return code                     +
               RSNCODE),            Reason code                     +
```

```
               VL,MF=(E,PLIST)       ---------------------------------
*  The return value (RETVAL) does not matter.  When your program
*  receives control following the call to cond_cancel, it is no
*  longer eligible to receive event notifications using cond_post.
```

Return Codes and Reason Codes

This service can return the following return code:

Return Code	Explanation
ECMSERR	The call was unsuccessful due to a CMS environmental or internal error.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “cond_setup (BPX1CSE) — Set Up to Receive Event Notifications” on page 50
- “cond_timed_wait (BPX1CTW) — Suspend a Thread for a Limited Time or for an Event” on page 52
- “cond_wait (BPX1CWA) — Suspend a Thread for an Event” on page 55.

cond_post (BPX1CPO) — Post a Thread for an Event

BPX1CPO

thread_ID
event
return_value
return_code
reason_code

Purpose

Use the cond_post (BPX1CPO) service to notify another thread in the process that an event has occurred.

Parameters

thread_ID
(input,CHAR,8) is a variable for specifying the thread ID for the thread that is to be notified of the event. The target thread must be in the same process as the caller.

event
(input,INT,4) is a variable for specifying an integer value that determines which event notification is to be sent to the target thread. This value represents an event for which the thread identified by *thread_ID* may be waiting. If the target thread is waiting, the service notifies it that the event has occurred.

The *event* parameter must be one of the following two event values, defined by the BPXYCW macro:

CW_CONDVAR
This value causes the target thread to resume processing if it is waiting for a CW_CONDVAR event.

CW_TIMEOUT
This value causes the target thread to resume processing if it is waiting for a timeout notification.

See "BPXYCW — Map Serialization Constants" on page 419.

Notes:

1. You must specify exactly one event.
2. Use of cond_post (BPX1CPO) to send a CW_TIMEOUT notification is restricted to programs that run in supervisor state with protect key 0.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. When the target thread is not cond_setup, cond_wait, or cond_timed_wait, cond_post does not post the thread and *return_value* is set to `0`.

2. The cond_post service attempts to send an event notification to the target thread. Event notifications are delivered to a target thread only when that thread is set up to receive them. A thread that is not set up to receive an event notification is discarded. The cond_post service does not check whether the target thread is set up to receive the event, so the cond_post service can return a value of 0 even though the event notification was discarded. Therefore, if you use the cond_wait and cond_post services to synchronize threads, you must be certain that the target thread is waiting for the event before you use cond_post to send the notification.

Characteristics and Restrictions

The target thread must be in the same process as the caller.

Example

The following code demonstrates how to send an event notification to a thread waiting in the cond_wait or cond_timed_wait service. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
The following code notifies thread (THID) that a CW_CONDVAR event
has occurred.
         CALL  BPX1CPO,              Send condition event notification +
               (THID,                Input: Thread ID of target pgm    +
               =A(CW_CONDVAR),       Input: Event            in BPXYCW +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EINVAL	The *thread_ID* parameter is not valid. It contains a value that is inconsistent with the thread IDs managed by the system. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRLightWeightThID, JRNoEvents, JRTooMany, JRUndefEvents.
ESRCH	The system determined that the *thread_ID* value does not refer to a thread that currently exists in the caller's process. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRThreadNotFound, JRAlreadyTerminated.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “cond_timed_wait (BPX1CTW) — Suspend a Thread for a Limited Time or for an Event” on page 52
- “cond_wait (BPX1CWA) — Suspend a Thread for an Event” on page 55.

cond_setup (BPX1CSE) — Set Up to Receive Event Notifications

```
BPX1CSE
        event_list
        return_value
        return_code
        reason_code
```

Purpose

Use the cond_setup (BPX1CSE) service to make the calling thread eligible to receive event notifications from other threads.

Parameters

event_list
(input,INT,4) is a variable for specifying a value that indicates which events are of interest to the thread. This value is the inclusive OR of one or more of the following event values, defined by the BPXYCW macro:

CW_INTRPT
The program running on the thread needs to know about signals sent to the thread.

CW_CONDVAR
The program running on the thread needs to suspend processing until some other thread uses the cond_post service to send this thread a notification of a CW_CONDVAR event.

Note: The C/C++ functions `pthread_cond_signal()` and `pthread_cond_broadcast()` use this value to send condition notifications.

You must specify at least one event; you may specify both. See "BPXYCW — Map Serialization Constants" on page 419.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. The scope of the effect of the cond_setup service is just until the next service is requested. The intended use of cond_setup is to set up for a subsequent call to cond_wait or cond_timed_wait. Other callable services that the program invokes between cond_setup and cond_wait or cond_timed_wait may fail with a return value of `-1`, a reason code of EINVAL, and a reason code of JRNotSetup.

   The only exception to this is the queue_interrupt service. You can use the queue_interrupt service to "put back" the last signal delivered to the signal interrupt routine.
2. If you use cond_setup to specify the events that cause the thread to resume processing, you must repeat the setup before each call to cond_wait or cond_timed_wait.

3. If you use cond_setup with cond_timed_wait, do not specify the CW_TIMEOUT condition on the call to cond_setup. The cond_timed_wait service provides setup for the CW_TIMEOUT event.
4. Calling the cond_setup servide before the cond_wait and cond_timed_wait services is optional.
5. If a thread has called cond_setup but has not called cond_wait or cond_timed_wait, any cond_post service to it are pending. When the cond_wait or cond_timed_wait service is called, the pending cond_post prevents the caller from waiting.

Characteristics and Restrictions

The program running on the thread should eventually call one of the cond_wait, cond_timed_wait, or cond_cancel services.

Example

The following code sets up the invoker to suspend processing until any of the specified events (CW_INTRPT or CW_CONDVAR) occurs. The BPX1CTW (cond_timed_wait) or BPX1CWA (cond_wait) service is used to actually suspend processing. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   EVENTLIST,=A(CW_INTRPT+CW_CONDVAR)
         CALL  BPX1CSE,              Condition setup                 +
               (EVENTLIST,           Input: Event list          BPXYCW +
               RETVAL,               Return value: 0 or -1           +
               RETCODE,              Return code                     +
               RSNCODE),             Reason code                     +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return code:

Return Code	Explanation
EINVAL	The system determined that the event list passed to the service is in error. Consult the reason code returned to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRAlreadySetup, JRNoEvents, JRUndefEvents.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “cond_cancel (BPX1CCA) — Cancel Interest in Events” on page 46
- “cond_post (BPX1CPO) — Post a Thread for an Event” on page 48
- “cond_timed_wait (BPX1CTW) — Suspend a Thread for a Limited Time or for an Event” on page 52
- “cond_wait (BPX1CWA) — Suspend a Thread for an Event” on page 55
- “queue_interrupt (BPX1SPB) — Return the Last Interrupt Delivered” on page 223.

cond_timed_wait (BPX1CTW) — Suspend a Thread for a Limited Time or for an Event

BPX1CTW

seconds
nanoseconds
event_list
seconds_remaining
nanoseconds_remaining
return_value
return_code
reason_code

Purpose

Use the cond_timed_wait (BPX1CTW) service to suspend the calling thread until any one of a set of events has occurred or until a specified amount of time has passed.

Parameters

seconds
(input,INT,4) is a variable for specifying an unsigned integer that is the maximum number of seconds the calling program is willing to wait for one of the specified events to occur.

Notes:

1. The *seconds* parameter can be any value from 0 to 4,294,967,295, inclusive.
2. The *seconds* and *nanoseconds* values are combined to determine the timeout value.

nanoseconds
(input,INT,8) is a variable for specifying an unsigned integer that is the number of nanoseconds to be added to the value specified by the *seconds* parameter.

Notes:

1. The *nanoseconds* parameter can be any value from 0 to 1,000,000,000, inclusive.
2. The *seconds* and *nanoseconds* values are combined to determine the timeout value.

event_list
(input,INT,4) is a variable for specifying a value that determines which events will cause the thread to resume processing.

The value contained in this variable is the inclusive OR of one or more of the following event values, defined by the BPXYCW macro:

CW_INTRPT
Suspends processing until a signal is sent to the thread. This is a cancellation point that is described in the usage notes of “pthread_setintr (BPX1PSI) — Examine and Change Interrupt State” on page 217.

CW_CONDVAR
Suspends processing until some other thread in the process sends this one a CW_CONDVAR notification.

See “BPXYCW — Map Serialization Constants” on page 419.

When the event list is zero, it means the caller has used the cond_setup service to specify the events, and the thread is already eligible to be notified of events. In this case, the cond_timed_wait service sets the timer for the specified interval and suspends thread processing until an event occurs, a signal arrives, or the time limit is reached.

seconds_remaining
(input/output,INT,4) is a variable where the service returns an unsigned value that is the number of seconds of unexpired time remaining in the time interval.

Note: This value is valid only when *return_value* is `0`, or when *return_value* is `-1` and the return code is EINTR.

nanoseconds_remaining
(input/output,INT,4) is a variable where the service returns an unsigned value that is the number of nanoseconds of unexpired time remaining in the time interval.

Notes:

1. The *nanoseconds_remaining* parameter can be any value from 0 to 1,000,000,000, inclusive.
2. This value is valid only when *return_value* is `0`, or when *return_value* is `-1` and the return code is EINTR.

return_value
(output,INT,4) is a variable where the service returns a `0` if a CW_CONDVAR event occurred, or `-1` otherwise.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. The cond_timed_wait service is substantially similar to the POSIX function `nanosleep()`. (Refer to the POSIX standard for a description of `nanosleep()`.) If you need the `nanosleep()` function, you can use cond_timed_wait to implement your own version.
2. If your program uses cond_timed_wait to wait for events that it specified by calling cond_setup, it must not call any other CMS services between the calls to cond_setup and cond_timed_wait. If the program invokes other callable services between cond_setup and cond_timed_wait, then the cond_timed_wait callable service fails with a return value of `-1`, a return code of EINVAL, and a reason code of JRNotSetup.

   The only exception to this is the queue_interrupt service. You can use the queue_interrupt service to "put back" the last signal delivered to the signal interrupt routine. A signal can arrive after the program running on the thread has called cond_setup and before it gets a chance to call cond_timed_wait. The program may choose to "put back" the signal to defer handling of it until a later time.
3. If you use cond_setup to specify the events that will cause the thread to resume processing, you must repeat the setup before each call to cond_wait or cond_timed_wait.
4. If you do not include the CW_INTRPT event when you use cond_timed_wait, some services used by other threads or processes cannot cause the waiting thread to resume processing. In particular, the following services do not cause an event notification unless CW_INTRPT is specified in the event list:
   - pthread_cancel
   - pthread_kill
   - pthread_quiesce
   - kill.

Characteristics and Restrictions

See Appendix E, “The Relationship of OpenExtensions Signals to Callable Services,” on page 557.

Example

The following code suspends the calling thread until a signal arrives (CW_INTRPT) or else 2.5 seconds have elapsed. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   EVENTLIST,=A(CW_INTRPT)          Signals
         CALL  BPX1CTW,              Wait for condition events         +
               (=A(2),               Input: Number of seconds          +
               =A(500000000),        Input: Number of nanoseconds      +
               EVENTLIST,            Input: Event list          BPXYCW +
               SECONDS,              Output: Unexpired seconds         +
               NANOSECONDS,          Output: Unexpired nanoseconds     +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EAGAIN	No signal or event notification arrived within the specified timeout period. The thread resumed processing because the time interval expired. **Note:** If you specify a value of 0 for both seconds and nanoseconds and no event notification is pending when you call cond_timed_wait, it returns this return code.
EINTR	A signal caused the cond_timed_wait service to resume processing of the thread. **Note:** The signal handler has already run.
EINVAL	The system determined that one or more of the parameters passed to the service are in error. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRAlreadySetup, JRNanoSecondsTooBig, JRNotSetup, JRUndefEvents.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “cond_cancel (BPX1CCA) — Cancel Interest in Events” on page 46
- “cond_post (BPX1CPO) — Post a Thread for an Event” on page 48
- “cond_setup (BPX1CSE) — Set Up to Receive Event Notifications” on page 50
- “cond_wait (BPX1CWA) — Suspend a Thread for an Event” on page 55
- “queue_interrupt (BPX1SPB) — Return the Last Interrupt Delivered” on page 223.

cond_wait (BPX1CWA) — Suspend a Thread for an Event

BPX1CWA

event_list
return_value
return_code
reason_code

Purpose

Use the cond_wait (BPX1CWA) service to suspend processing on the calling thread until any one of a set of events has occurred.

Parameters

event_list
(input/output,INT,4) is a variable for specifying a value that determines which events will cause the thread to resume processing.

This value is the inclusive OR of one or more of the following event values, defined by the BPXYCW macro:

CW_INTRPT
Suspends processing until a signal is sent to the thread.

CW_CONDVAR
Suspends processing until some other thread in the process sends this one a CW_CONDVAR event notification.

See "BPXYCW — Map Serialization Constants" on page 419.

An *event_list* value of 0 means the caller has used the cond_setup (BPX1CSE) service to specify the events, and the thread is already eligible to be notified of events. In this case, the cond_wait (BPX1CWA) service suspends thread processing until an event occurs or a signal arrives.

return_value
(output,INT,4) is variable where the service returns a 0 if a CW_CONDVAR event occurred, or -1 otherwise.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. If your program uses cond_wait to wait for events that it specified by calling cond_setup, it must not call any other CMS services between the calls to cond_setup and cond_wait. If the program invokes other callable services between cond_setup and cond_wait, the cond_wait callable service fails with a return value of -1, a return code of EINVAL, and a reason code of JRNotSetup.

   The only exception to this is the queue_interrupt service. You may use the queue_interrupt service to "put back" the last signal delivered to the signal interrupt routine. A signal may arrive after the program running on the thread has called cond_setup and before it gets a chance to call cond_wait. The program may choose to "put back" the signal to defer handling it until a later time.

If you use cond_setup to specify the events that will cause the thread to resume processing, you must repeat the setup before each call to cond_wait or cond_timed_wait.

2. If you do not include the CW_INTRPT event when you use cond_wait, some services used by other threads or processes cannot cause the waiting thread to resume processing. In particular, the following services do not cause an event notification unless CW_INTRPT is specified in the event list:
   - pthread_cancel
   - pthread_kill
   - pthread_quiesce
   - kill

Characteristics and Restrictions

See Appendix E, “The Relationship of OpenExtensions Signals to Callable Services,” on page 557.

Example

The following code suspends the calling thread until either of two events occurs. The two events are the arrival of a signal (CW_INTRPT) or some other thread using the cond_post (BPX1CPO) service to send this thread a CW_CONDVAR notification. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   EVENTLIST,=A(CW_INTRPT+CW_CONDVAR)
         CALL  BPX1CWA,              Wait for condition events         +
               (EVENTLIST,           Input: Event list          BPXYCW +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EINTR	A signal caused the cond_wait service to resume processing of the thread. **Note:** The signal handler has already run.
EINVAL	The system determined that one or more of the parameters passed to the service are in error. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRAlreadySetup, JRNotSetup, JRUndefEvents.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “cond_cancel (BPX1CCA) — Cancel Interest in Events” on page 46
- “cond_post (BPX1CPO) — Post a Thread for an Event” on page 48
- “cond_setup (BPX1CSE) — Set Up to Receive Event Notifications” on page 50
- “cond_timed_wait (BPX1CTW) — Suspend a Thread for a Limited Time or for an Event” on page 52
- “queue_interrupt (BPX1SPB) — Return the Last Interrupt Delivered” on page 223.

connect (BPX1CON) — Establish a Connection Between Two Sockets

BPX1CON

socket_de scriptor
sockaddr_length
sockaddr
return_value
return_code
reason_code

Purpose

For stream sockets, use the connect (BPX1CON) service to establish a connection from a client socket to a socket at a server. For datagram sockets, use the connect service to specify the peer for a socket.

Stream sockets can call the connect service only once. Datagram sockets can call the connect service multiple times to change their association.

Parameters

socket_descriptor
(input,INT,4) is a variable for specifying the descriptor of the socket to be connected.

sockaddr_length
(input,INT,4) is a variable for specifying the length of the *sockaddr* parameter.

sockaddr
(input,INT,*sockaddr_length*) is a variable for specifying the SOCKADDR structure that contains the address of the socket or the name of the peer to which a connection is to be attempted. The SOCKADDR structure is mapped by the BPXYSOCK macro. See "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. For connectionless sockets, the connect service may be advantageous because the destination address need not be specified for every datagram sent. Once a UDP (connectionless) socket is connected, the read, write, recv, send, readv, and writev system calls can be used for I/O on those sockets. Otherwise, only the sendto/recvfrom and sendmsg/recvmsg system calls can be used. Once a UDP socket is connected, only datagrams from the specified *sockaddr* are received on the socket. To disconnect a UDP socket from a previous connection, issue the connect system call with an incorrect *sockaddr*, such as a null address.

2. The connect callable service can be used to test whether a target socket is available for the connect. If the socket is not available, an ECONNREFUSED is returned.

Example

The following code connects to a socket. SOCKDESC was returned by a previous call to socket (BPX1SOC), and SOCKADDR contains the name of the peer, possibly obtained by a call to getpeername (BPX1GNM). This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services” on page 465.

```
         SPACE ,
         CALL  BPX1CON,              Connect to a socket              +
               (SOCKDESC,            Input: Socket Descriptor         +
               SOCK#LEN+SOCK_SUN#LEN, Input: Length - Sockaddr        +
               SOCKADDR,             Input: Sockaddr structure        +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EADDRNOTAVAIL	The specified address is not available from the local machine.
EAFNOSUPPORT	The address family that was specified in the address structure is not supported.
EALREADY	The socket descriptor *socket* is marked nonblocking, and a previous connection attempt has not completed.
EBADF	The socket descriptor is incorrect. The following reason codes can accompany this return code: JRFileDesNotInUse, JRFileNotOpen.
ECONNREFUSED	The attempt to connect was rejected. The following reason code can accompany this return code: JRListenNotDone.
EINPROGRESS	O_NONBLOCK is set for the file descriptor for the socket, and the connection cannot be immediately established. The connection will be established asynchronously. The EINPROGRESS value does not indicate an error condition.
EINTR	A signal interrupted the connect service before this connection was accepted. The following reason code can accompany this return code: JRSignalReceived.
EINVAL	One of the input parameters is not correct. The following reason codes can accompany this return code: JRSocketCallParmError, JRSockNoName.For AF_UNIX, the following reason codes can accompany this return code: JRRdlBufflenInvalid, JRFileNotExtLink
EIO	An I/O error occurred. The following reason code can accompany this return code: JRPrevSockError.
EISCONN	The socket is already connected.
ENETUNREACH	For an AF_INET or AF_INET6 connection, this indicates that the network cannot be reached from this host. For an AF_IUCV connection, this means that the specified user ID is not logged on or is no longer accepting connections. For an AF_UNIX connection, this means that the user ID that created the specified path name is not logged on or is no longer accepting connections.

Return Code	Explanation
ENOBUFS	A buffer could not be obtained.
ENOTSOCK	*socket_descriptor* does not refer to a valid socket descriptor. The following reason code can accompany this return code: JRMustBeSocket.
EOPNOTSUPP	The socket is ready to accept connections. An accept request was expected. The following reason code can accompany this return code: JRListenAlreadyDone.
EPROTOTYPE	The address specifies a socket that is not the correct type for this request. The following reason code can accompany this return code: JRIncorrectSocketType.
EWOULDBLOCK	The socket is marked nonblocking, and the connection cannot be completed immediately.

The following are for AF_UNIX only:

Return Code	Explanation
EACCES	The process does not have search permission on a component of the path prefix, or it does not have write access to the named socket.
EIO	An I/O error occurred while reading from or writing to the file system.
ELOOP	Too many symbolic links were encountered in translating the path name in *sockaddr*.
ENAMETOOLOG	A component of a path name exceeded NAME_MAX characters, or an entire path name exceeded PATH_MAX characters.
ENOENT	The AF_UNIX path name is not valid. The following reason code can accompany the return code: JRFileNotThere
ENOTDIR	A component of the path prefix of the path name in *sockaddr* is not a directory.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

create_external_link (BPX1ELN) — Create a CMS External Link

BPX1ELN

link_contents_length
link_contents
link_name_length
link_name
mode
return_value
return_code
reason_code

Purpose

Use the create_external_link (BPX1ELN) service to create a CMS external link. An external link can be used to:

- Reference data outside of BFS (data residing on a CMS minidisk or in an SFS directory)
- Create an implicit mount point
- Contain data in an application-defined format

Parameters

link_contents_length
(input,INT,4) is a variable for specifying the length of the *link_contents* parameter. This must be value between 1 and 1023.

link_contents
(input,CHAR,*link_contents_length*) is a variable for specifying the contents of the external link to be created. The format of the information provided in this parameter depends on the file subtype specified in the *mode* parameter. For detailed information on the file subtypes, see the usage notes.

link_name_length
(input,INT,4) is a variable for specifying the length of the *link_name* parameter. This must be value between 1 and 1023.

link_name
(input,CHAR,*link_name_length*) is a variable for specifying the name of the external link being created.

mode
(input,INT,4) is a variable for specifying the mode of the external link. The mode includes the file type, the file subtype, and the permissions you grant to yourself, to your group, and to any user.

The file type and subtype are identified using the BPXYFTYP macro. Permissions are specified with the BPXYMODE macro. See "BPXYFTYP — Map File Type Definitions" on page 423 and "BPXYMODE — Map Mode Constants" on page 437. File subtypes are described in detail in the usage notes.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. The BPX1ELN service creates external links, which are BFS objects. External links have different functions based on their subtype. The subtypes are:

   FST_EXEC
   The external link identifies a CMS module file that resides on a minidisk or in an SFS directory. The file is executed when specified on the invocation of the:

   - C-Language `exec()` system call or the exec (BPX1EXC) callable service
   - C-Language `spawn()` system call or the spawn (BPX1SPN) callable service
   - OPENVM RUN command

   FST_DATA
   The external link identifies a non-BFS file residing on a minidisk or in an SFS directory that can be accessed by C-Language calls, such as `open()`, `read()`, `write()`, `close()`, and so on.

   FST_MEL
   The external link identifies a Mount External Link (MEL). When a MEL is encountered during path name resolution, it is treated as a directory with a file system mounted on it. Path name resolution continues in the "mounted" file system.

   FST_SOCKET
   (Reserved for IBM Use Only) The external link identifies a socket. Sockets external links are created through appropriate C library functions; they should not be explicitly created by user-written applications.

   User Defined
   Subtypes in the range of 100-200 (decimal) are reserved for application-defined external links.
2. You can also create external links by using the CMS OPENVM CREATE EXTLINK command. For more information, see the *z/VM: OpenExtensions Commands Reference*.
3. No syntax verification is done on the content of a link when an external link is created. The syntax is verified by the individual functions that refer to the external link.
4. When FST_DATA and FST_EXEC external links are used, authorization checking is done on two levels. The first authorization is verified based on the mode parameter associated with the external link. This authorization is done according to the POSIX requirements. The second authorization is based on the traditional CP and CMS authorization rules for linking and accessing minidisks, SFS directories and files, and so on.
5. An FST_DATA external link specified on a C-language `open()` system call is converted into an ANSI `fopen()` internally by the C run-time library. The access mode used for the `fopen()` is coded in the external link contents as &&& or &&B (or &&b). The characters && are replaced with the access mode specified on the `open()` request according to Table 2 on page 61.

Table 2. open() Request Access Mode with Characters &&

Access mode on fopen()	Access mode on open()	
	with &&&	with &&B
O_RDONLY	r	rb
O_WRONLY	r+	r+b
O_RDWR	r+	r+b
O_WRONLY + O_APPEND	a	ab
O_RDWRLY + O_APPEND	a+	a+b

Table 2. open() Request Access Mode with Characters && (continued)		
Access mode on fopen()	**Access mode on open()**	
	with &&&	**with &&B**
O_WRONLY + O_TRUNC	w	wb
O_RDWRLY + O_TRUNC	w+	w+b
O_WRONLY + O_APPEND + O_TRUNC	-	-
O_RDWR + O_APPEND + O_TRUNC	-	-
Note: O_WRONLY is not strictly supported; it is mapped to O_RDWR. The O_CREAT, O_EXCL, O_NOCTTY, and O_NONBLOCK flags are ignored.		

Example

The following examples follow the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551.

Example 1: The following code creates an external link named /u/dpt37/payroll with the subtype FST_DATA. Specifying this external link on a C-program `open()` system call is like specifying the link contents on a C-program `fopen()` system call.

```
MVC   BUFFERA(20),=CL20'//PAYROLL.FILE.A,&&&'
MVC   BUFLENA,=F'20'
MVC   BUFFERB(16),=CL16'/u/dpt37/payroll'
MVC   BUFLENB,=F'16'
XC    S_MODE,S_MODE
MVI   S_TYPE,FT_EXTLINK     External Link
MVI   S_SUBTYPE,FST_DATA    sub-type: DATA
MVI   S_MODE2,S_IRUSR       Read Wrt Read Wrt Read
MVI   S_MODE3,S_IWUSR+S_IRGRP+S_IWGRP+S_IROTH
SPACE ,
CALL  BPX1ELN,              Create external link             +
      (BUFLENA,             Input: Link contents length      +
      BUFFERA,              Input: Link contents             +
      BUFLENB,              Input: Link name length          +
      BUFFERB,              Input: Link name                 +
      MODE,                 Input: Mode                      +
      RETVAL,               Return value: 0 or -1            +
      RETCODE,              Return code                      +
      RSNCODE),             Reason code                      +
      RSNCODE),             Reason code                      +
      VL,MF=(E,PLIST)       ---------------------------------
```

If you later run a C program which has the following statements in it:

```
.....
fd = open("/u/dpt37/payroll",O_RDWR);
read(fd,buffA,n);
write(fd,buffB,m);
.....
```

the result would be as if you executed:

```
.....
FILE * stream;
stream = fopen("//PAYROLL.FILE.A","r+ ");
fread(buffA,1,n,stream);
fwrite(buffB,1,m,stream);
.....
```

Example 2: The following code creates an external link named /clearscreen with subtype FST_EXEC. Specifying this external link on the CMS OPENVM RUN command (or on a C-program `exec()` or `spawn()`

system call) causes the VMFCLEAR MODULE, loaded as a nucleus extension or residing on an accessed minidisk or SFS directory, to be executed.

```
         MVC   BUFFERA(17),=CL17'VMFCLEAR MODULE *'
         MVC   BUFLENA,=F'17'
         MVC   BUFFERB(16),=CL16'/bin/clearscreen'
         MVC   BUFLENB,=F'16'
         XC    S_MODE,S_MODE
         MVI   S_TYPE,FT_EXTLINK     External Link
         MVI   S_SUBTYPE,FST_EXEC    sub-type: EXECUTABLE
         MVI   S_MODE3,S_IXUSR+S_IXGRP+S_IXOTH
         SPACE ,
         CALL  BPX1ELN,              Create external link             +
               (BUFLENA,             Input: Link contents length      +
               BUFFERA,              Input: Link contents             +
               BUFLENB,              Input: Link name length          +
               BUFFERB,              Input: Link name                 +
               MODE,                 Input: Mode                      +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Example 3: The following code creates an external link named /u/gene with subtype FST_MEL. References to this external link will cause path name resolution to continue at the directory identified by the contents of the external link (/../VMBFS:VMSYSU:EUGENE/work/).

```
         MVC   BUFFERA(17),=CL29'/../VMBFS:VMSYSU:EUGENE/work/'
         MVC   BUFLENA,=F'29'
         MVC   BUFFERB(16),=CL17'/u/gene'
         MVC   BUFLENB,=F'7'
         XC    S_MODE,S_MODE
         MVI   S_TYPE,FT_EXTLINK     External Link
         MVI   S_SUBTYPE,FST_MEL     sub-type: MOUNT
         MVI   S_MODE2,S_IRUSR       Read Wrt Srch Read Srch Read Srch
         MVI   S_MODE3,S_IWUSR+S_IXUSR+S_IRGRP+S_IXGRP+S_IROTH+S_IXOTH
         SPACE ,
         CALL  BPX1ELN,              Create external link             +
               (BUFLENA,             Input: Link contents length      +
               BUFFERA,              Input: Link contents             +
               BUFLENB,              Input: Link name length          +
               BUFFERB,              Input: Link name                 +
               MODE,                 Input: Mode                      +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The requested operation requires writing in a directory with a mode that denied write permission.
EEXIST	The external link already exists. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRExtFileAlreadyExists.
EINVAL	At least one parameter is not valid. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRCompNotDir, JREndingSlashExtlink, and JRInvalidExtLinkLen.

Return Code	Explanation
ENAMETOOLONG	The *link_name* argument is longer than 1023 characters, or some component of that name is longer than 255 characters. CMS does not support name truncation.
EROFS	The requested operation requires writing in a directory on a read-only file system. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRReadOnlyFS.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “chown (BPX1CHO) — Change the Owner or Group of a File or Directory” on page 31
- “exec (BPX1EXC) — Run a Program” on page 72
- “lstat (BPX1LST) — Get Status Information about a File or Symbolic Link by Path Name” on page 157
- “mkdir (BPX1MKD) — Make a Directory” on page 160
- “mknod (BPX1MKN) — Make a FIFO or Character Special File” on page 163
- “mount (BPX1MNT) — Make a File System Available” on page 166
- “open (BPX1OPN) — Open a File” on page 181
- “readlink (BPX1RDL) — Read the Value of a Symbolic Link” on page 236
- “rename (BPX1REN) — Rename a File or Directory” on page 251
- “rmdir (BPX1RMD) — Remove a Directory” on page 256
- “spawn (BPX1SPN) — Spawn a Process” on page 333
- “symlink (BPX1SYM) — Create a Symbolic Link to a Path Name” on page 345
- “unlink (BPX1UNL) — Remove a Directory Entry” on page 379.

create_thread_environment (BPX1CTE) — Create POSIX Thread Environment

BPX1CTE

module_name
return_value
return_code
reason_code

Purpose

Use the create_thread_environment (BPX1CTE) service to create the language environment necessary to support POSIX threads.

Parameters

module_name
(input,CHAR,8) is a variable for specifying the name of the language exits module. The name must be left justified and padded with blanks.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Note

The language exits module should be built in the same way as described for the language environment manager in the *z/VM: CMS Application Multitasking*.

Example

The code in this example initializes the POSIX process environment and establishes the assembler language environment manager. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
MVC   LANGMAN(8),=C'DMSHASM '   Assembler Environment Manager
SPACE ,
CALL  BPX1CTE,                                          +
      (LANGMAN,             Input: Language Manager     +
      RETVAL,               Return value: -1 or not return +
      RETCODE,              Return code                 +
      RSNCODE),             Reason code                 +
      VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECMSINITIAL	The maximum number of OpenExtensions processes has already been reached.
ENOENT	The specified language exits module cannot be found or loaded
ENOMEM	There is not enough storage to load the specified language exits module.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

DLL_delete (BPX1DEL) — Delete a Program from Storage

BPX1DEL

entrypt_address

return_value

return_code

reason_code

Purpose

Use the DLL_delete (BPX1DEL) service to delete a previously-loaded program from the storage of the caller's process.

Parameters

entrypt_address
(input,INT,4) is a variable for specifying the entry point address of the program to be deleted. This value was returned by the DLL_load (BPX1LOD) service when the program was loaded.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. Calling DLL_delete (BPX1DEL) to delete a program from storage may not actually result in the program being removed from storage. If the program has been loaded more than once, the program remains in storage until DLL_delete is called the exact number of times that the program was loaded.

Example

The program ictasma located at **ict/bin** is loaded into storage using BPX1LOD, branched to, and then deleted from storage using BPX1DEL.

```
         MVC   BUFLENA,=F'16'
         MVC   BUFFERA(16),=C'/ict/bin/ictasma'
         MVC   OPTIONS,=A(0)
         MVC   LIBPTHLN,=A(0)
         SPACE ,
         CALL  BPX1LOD,              Load Program                      +
               (BUFLENA,             Input: Pathname length            +
               BUFFERA,              Input: Pathname                   +
               OPTIONS,              Input: Options                    +
               LIBPTHLN,             Input: Library Path Length        +
               LIBPATH,              Input: Library Path               +
               EPADDR,               Return value: -1 or entry pt addr +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
         L     R15,EPADDR            Load return value
         C     R15,=F'-1'            Test for -1 return
         BE    PSEUDO                Branch on error
         SPACE ,
```

```
         L     R15,EPADDR
         BALR  R14,R15               Branch to loaded program
         SPACE ,
         CALL  BPX1DEL,              Delete program                    +
               (EPADDR,              Input: Entry point address        +
               RETVAL,               Return value: -1 or 0             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EINVAL	The *entrypt_address* parameter contains an entry point address that is not valid. The specified entry point address does not represent a currently loaded program in the caller's process.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Another callable service related to this service is:

- “DLL_load (BPX1LOD) — Load a Program into Storage” on page 69

DLL_load (BPX1LOD) — Load a Program into Storage

BPX1LOD

filename_length
filename
flags
libpath_length
libpath
return_value
return_code
reason_code

Purpose

Use the DLL_load (BPX1LOD) service to load an executable program into the caller's process.

Parameters

filename_length
(input,INT,4) is a variable for specifying the length of the *filename* parameter. The length can be a value in the range 1 to 1023.

filename
(input,CHAR,*filename_length*) is a variable for specifying the name of the file to be loaded:

- If *filename* does not contain a / (slash), it is treated as a base name, and should be in one of the directories listed in the supplied *libpath* parameter. If the *libpath* parameter is null, the file must be in the current directory.
- If *filename* is not a base name (it contains at least one / (slash)), the name is used "as is" without using the *libpath* parameter to locate the file.
- If *filename* is a base name, it can be up to 255 characters long.
- If *filename* is a path name, see “Understanding Byte File System (BFS) Path Name Syntax” on page 6.

flags
(input,INT,4) is a variable for specifying option flags that indicate what optional processing is to be performed on behalf of the caller. The only valid values for this parameter are: X'00000000'.

libpath_length
(input,INT,4) is a variable for specifying the length of the *libpath* parameter. If the value of this parameter is zero, the *libpath* parameter is ignored.

libpath
(input,CHAR,*libpath_length*) is a variable for specifying the library path to be searched in determining the fully-qualified path name of the file specified in the *filename* parameter. The library path consists of a series of path names separated by colons. The path names in the list are searched one at a time until the specified file name is located. If the list of path names begins with a colon or ends with a colon, the working directory of the calling process is used to locate the file. Each path name in the list can have a maximum length of 1024 bytes.

The following is an example of a valid library path:

```
/usr1/bin:/grp1/bin:/bin
```

return_value
(output,INT,4) is a variable where the service returns the entry point address of the program that was loaded into storage if the request is successful, or -1 if the request is not successful.

If the loaded program is an AMODE(31) program, the high order bit of the return value is turned on. For this reason, applications testing for a failure condition must explicitly check for a -1 value. Simply checking for a value of less than zero will not produce the desired results.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. If the specified file name represents an external link, the program is loaded from the caller's CMS search order. The external name is used only if the name is eight characters or less; otherwise, the caller receives an error from the DLL_load service.
2. When running from a pthread_created thread (pthread) the specified file is loaded into storage and associated with the Initial Pthread Creating Task (IPT) to allow the sharing of a program across multiple threads without the problem of the program disappearing unexpectedly when a thread terminates.
3. Because this service does not cause the specified program to be executed, the set-user-ID and set-group-ID flags have no impact on the process.
4. If a program that is loaded into storage with this service is not deleted from storage, then the program remains in storage until the calling task terminates, if it is not a pthread, or when the Initial Pthread Creating Task (IPT) terminates, if the caller is a pthread.

Characteristics and Restrictions

There are no restrictions on the use of DLL_load.

Example

The program ictasma located at **ict/bin** is loaded into storage and then branched to.

```
         MVC   BUFLENA,=F'16'
         MVC   BUFFERA(16),=C'/ict/bin/ictasma'
         MVC   OPTIONS,=A(0)
         MVC   LIBPTHLN,=A(0)
         SPACE ,
         CALL  BPX1LOD,              Load program                      +
               (BUFLENA,             Input: Pathname length            +
               BUFFERA,              Input: Pathname                   +
               OPTIONS,              Input: Options                    +
               LIBPTHLN,             Input: Library Path Length        +
               LIBPATH,              Input: Library Path               +
               EPADDR,               Return value: -1 or entrypt addr  +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
         SPACE ,
         L     R15,EPADDR            Load return value
         C     R15,=F'-1'            Test for -1 return
         BE    PSEUDO                Branch on error
         SPACE ,
         L     R15,EPADDR
         BALR  R14,R15               Branch to loaded program
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The caller does not have appropriate permissions to run the specified file. It may lack permission to search a directory named in the *pathname* parameter; or it may lack execute permission for the file to be run; or the file to be run is not a regular file, and the system cannot run files of that type.
ENAMETOOLONG	The *filename* argument is longer than 1023 characters, or some component of the file name is longer than 255 characters. CMS does not support name truncation.
ENOENT	No file name was specified, or one or more of the components of the specified *filename* were not found.
ENOEXEC	The specified file has execute permission, but is not in the proper format to be a process image file.
ENOMEM	The file to be loaded requires more memory than is permitted by the hardware or the operating system.
ENOTDIR	A directory component of *filename* is not a directory.
EINVAL	The *flags* parameter specified contains an unsupported value.
EMFILE	Too many open files. An attempt was made to open more than the maximum number of file descriptors (OPEN_MAX) allowed in this process.
ENFILE	Too many files are open in the system. The system reached its predefined limit for simultaneously open files and temporarily could not accept requests to open another one.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Another callable service related to this service is:

- “DLL_delete (BPX1DEL) — Delete a Program from Storage” on page 67

exec (BPX1EXC) — Run a Program

BPX1EXC

pathname_length
pathname
argument_count
argument_length_list
argument_list
environment_count
environment_data_length
environment_data_list
exit_routine_address
exit_parameter_list_address
return_value
return_code
reason_code

Purpose

Use the exec (BPX1EXC) service to run a CMS module file. You identify the file by its path name. This service replaces the current process image that calls the service with a new process image for the executable file being run.

Parameters

pathname_length
(input,INT,4) is a variable for specifying the length of the *pathname* parameter.

pathname
(input,CHAR,*pathname_length*) is a variable for specifying the path name of the file to be run. See "Understanding Byte File System (BFS) Path Name Syntax" on page 6.

The name specified in this parameter is case-sensitive (not automatically uppercased), whether the file resides in BFS or outside of BFS. For information on how the exec service searches for the specified file, see usage notes "11" on page 74 and "12" on page 74.

argument_count
(input,INT,4) is a variable for specifying the number of 4-byte pointers in the arrays specified in the *argument_length_list* and *argument_list* parameters. If the program needs no arguments, specify 0.

argument_length_list
(input,INT,*argument_count*) is a variable for specifying an array of 4-byte pointers, each of which is the address of a fullword containing the length of an argument to be passed to the specified program. If the program needs no arguments, specify 0.

argument_list
(input,INT,*argument_count*) is a variable for specifying an array of 4-byte pointers, each of which is the address of a character string to be passed to the specified program as an argument. The length of each argument is specified by the corresponding element in the *argument_length_list* parameter. If the program needs no arguments, specify 0.

environment_count
(input,INT,4) is a variable for specifying the number of 4-byte pointers in the arrays specified in the *environment_data_length* and *environment_data* parameters. If the program needs no environment data, specify `0`.

environment_data_length
(input,INT,*environment_count*) is a variable for specifying an array of 4-byte pointers, each of which is the address of a fullword containing the length of an environment variable to be passed to the specified program. If the program does not use environment variables, specify `0`.

environment_data_list
(input,INT,*environment_count*) is a variable for specifying an array of 4-byte pointers, each of which is the address of a character string to be passed to the specified program as an environment variable. The length of each environment variable is specified by the corresponding element in the *environment_data_length* parameter. If the program does not use environment variables, specify `0`.

exit_routine_address
(input,INT,4) is a variable for specifying the address of the user's exit routine. If a user exit is not to be invoked, specify `0`.

exit_parameter_list_address
(input,INT,4) is a variable for specifying the address of the user exit parameter list. This value is in register 1 when the user exit receives control. If the user exit is not to be invoked or does not require parameters, specify `0`.

return_value
(output,INT,4) is a variable where the service returns `-1` if it is not successful. If successful, the service does not return.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. This call is not supported from REXX. If BPX1EXC is called from REXX, results are unpredictable. If you want to invoke a BFS file from REXX, consider using the spawn (BPX1SPN) service.
2. The following characteristics of the calling process are changed when the service gives control to the new executable file:
   - The current process image is replaced with a new process image for the executable file to be run.
   - All open file descriptors remain open unless the FCTLCLOEXEC flag is set.
   - Signals set to be caught are reset to their default.
   - If the set-user-ID mode bit of the new executable file is set and the invoker is authorized, the effective user ID and saved set-user-ID of the process are set to the owner user ID of the new executable file. See “BPXYMODE — Map Mode Constants” on page 437. The effective user ID of the process is always saved as the saved set-user-ID.
   - If the set-group-ID mode bit of the new executable file is set and the invoker is authorized, the effective group ID and saved set-group-ID of the process are set to the group ID of the new executable file. See “BPXYMODE — Map Mode Constants” on page 437. The effective group ID of the process is always saved as the saved set-group-ID.
3. The S_ISVTX (sticky) mode bit of the executable file is not supported by OpenExtensions and is ignored.
4. The new process image inherits the following from the calling process image:
   - Process ID
   - Parent process ID

- The time left until an alarm signal is generated
- File mode creation mask
- Process signal mask
- Pending signals
- Time accounting information.

For more information, see "times (BPX1TIM) — Get Process and Child Process Times" on page 371 and "BPXYTIMS — Map the Processor Time Structure for the times Service" on page 475.

5. All open files and directories that are not in the byte file system will remain open.
6. There is no return to the caller on a successful invocation of the exec service. Any storage subpools associated with the svc level of the caller are released.
7. The register usage on entry to the user exit is:
   - R0: Undefined.
   - R1: Address of the user exit parameter list, as specified by the caller of the exec service.
   - R2–R12: Undefined.
   - R13: Address of a 96-byte work area in user storage.
   - R14: The return address from the user exit to the exec service. This address *must* be preserved by the user exit.
   - R15: Address of the user exit.
8. The user exit receives control with the following attributes:
   - Supervisor state
   - PSW key of the invoker of exec
   - Amode=31
   - Enabled for interrupts
9. BFS authorization checking is performed on the module to be executed. The file permissions must specify execute authority for the file class to which the caller belongs (file owner class, file group class or file other class).
10. If the file to be executed resides in a file pool that is accessed through TSAF or AVS, it cannot be invoked if either the set-group-ID mode bit or the set-user-ID mode bit is on and either the effective UID or the effective GID of the caller does not match that of the file.
11. Unlike the C/C++ `execlp()` and `execvp()` functions, BPX1EXC does not use the environment variable PATH to construct a search order.
12. The file to be invoked must be a relocatable executable CMS module created by the GENMOD command, the BIND command, the c89 utility, or the `cxx` utility. The file type does not have to be MODULE. (If the file is not relocatable, results are unpredictable.)

    The file can reside in the byte file system or in the CMS record file system. The exec service first looks for an executable file in the byte file system. If this fails, the service looks for an external link with a subtype of FST_EXEC. If the file is not an external link, the service parses the path name into a CMS file ID and looks for the file in the record file system.

    If the file is an external link or a CMS file ID and the file type is not specified, MODULE is assumed. If the file mode is not specified, * is assumed. If the file type is MODULE or *, and the file mode is *, the exec service searches for a nucleus extension.

    To ensure that a nucleus extension is run in the calling process, it must have been established in the CMS Commands process or in the same CMS process that invokes this service. If a nucleus extension is run in a process other than the calling process, and it uses OpenExtensions services, results are unpredictable.

If the file is not a nucleus extension, or no search was made for a nucleus extension because the file ID criteria described above were not met, the exec service then searches for the file on the accessed minidisks and directories.

13. If the CMS module file to be executed contains MAP information, it is copied into the loader tables. However, because the loader tables are shared among all the processes in the virtual machine, the information in the loader tables cannot safely be relied upon in a multitasking environment.
14. The executable file to be run receives control with the same attributes as if it were invoked by CMSCALL, except that register 1 contains the address of an exec style parameter list, and the contents of register 0 are not defined. The parameter list consists of the following parameter addresses. In the last parameter address, the high-order bit is 1.

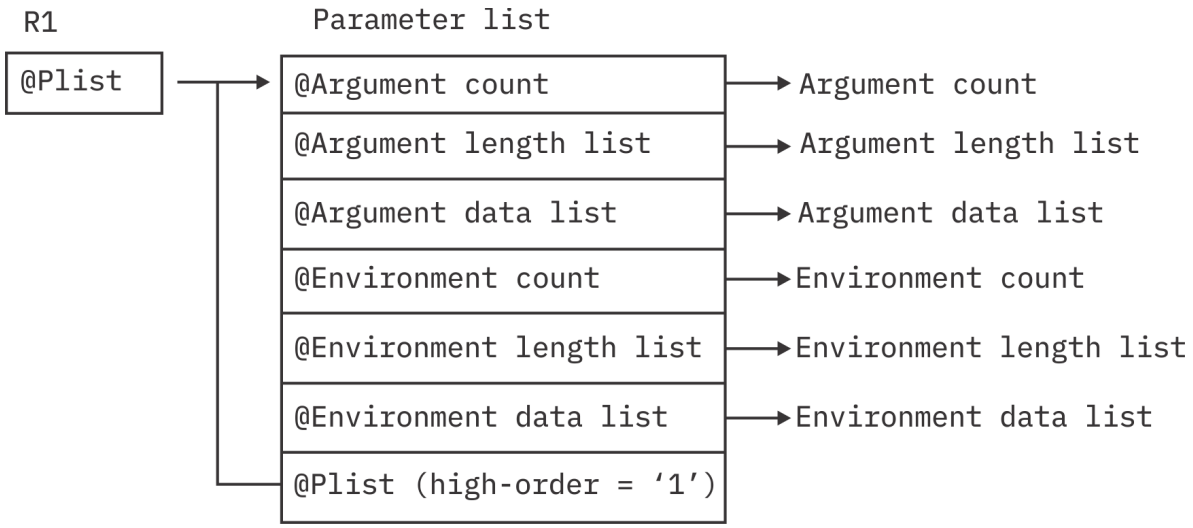


Figure 2. Parameter list passed to the module

The last parameter passed to the executable file identifies the parameter list as a POSIX style parameter list.

Register 13 contains a pointer to a user save area that you can use to save the calling program's registers. Note, however, that saving the caller's registers is optional, because CMS does it automatically. The user save area also contains a call type flag (USECTYP) that is set to X'10' to indicate that register 1 points to an exec style parameter list. The user save area can be mapped using the USERSAVE macro.

15. Exec performs preliminary error checking before removing the caller from storage. Control will return to the caller if an error is detected at this time. If an error is encountered after the caller is cleaned up, an abend will occur.
16. If the set-user-ID or set-group-ID mode bit of the executable file is set and will result in a change to the effective user ID or effective group ID, then the requestor must be authorized to have its IDs changed, and the file server on which the file resides must be authorized to change the IDs of another user.

    The following authorization applies to the requestor:

    - The External Security Manager (ESM) must grant the requestor authority to have its IDs changed, or
    - An ESM must not be installed or must defer authorization to CP, and:
      - The effective UID of the active process must be 0, or
      - The requesting VM user ID must have the attribute POSIXOPT EXEC_SETIDS ALLOW set, either through a statement in its CP directory entry or through a specified or defaulted setting in the system configuration file that is not overridden in the directory entry.

    The following authorization applies to the file server on which the file resides:

- The ESM must have identified to CP that the file server is authorized to change the IDs of another user when the file server logged on, or
- An ESM must not be installed or must defer authorization to CP, and the file server must have the attribute POSIXOPT SETIDS ALLOW set through a statement in its CP directory entry.

Characteristics and Restrictions

The user exit is given control while the exec (BPX1EXC) service is still in progress. The user exit should not attempt to use any OpenExtensions service that alters or terminates the current process (that is, the exec, exit, and kill services). If such services are attempted, the results are unpredictable. Signals cannot be delivered while in the user exit, because the exec service is still in progress and signal delivery is inhibited.

Example

The program ictasma located at **ict/bin** gets control and is passed arguments WK18, DEPT37A, and RATE(STD,NOEXC,NOSPEC). No environment arguments are passed. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   BUFLENA,=F'15'
         MVC   BUFFERA(15),=C'ict/bin/ictasma'
         MVC   ARGCNT,=F'3'
*                                First
         LA    R15,4                  Length
         ST    R15,ARGLLST+00         Length parm list
         LA    R15,=CL4'WK18'         Argument
         ST    R15,ARGSLST+00         Argument address parm list
*                                Second
         LA    R15,7                  Length
         ST    R15,ARGLLST+04         Length parm list
         LA    R15,=CL7'DEPT37A'      Argument
         ST    R15,ARGSLST+04         Argument address parm list
*                                Third
         LA    R15,22                 Length
         ST    R15,ARGLLST+08         Length parm list
         LA    R15,=CL22'RATE(STD,NOEXC,NOSPEC)'    Argument
         ST    R15,ARGSLST+08         Argument address parm list
*
         MVC   ENVCNT,=F'0'          Number of env. data items passed
         MVC   ENVLENS,=F'0'         Addr of end. data length list
         MVC   ENVPARMS,=F'0'        Add of env. data
*
         MVC   EXITRTNA,=V(EXITRTN)  ->exit routine
*        MVC   EXITPLA,=A(exit parameter list as expected by EXITRTN)
         SPACE ,
         CALL  BPX1EXC,                                          +
               (BUFLENA,             Input: Pathname length      +
               BUFFERA,              Input: Pathname             +
               ARGCNT,               Input: Argument count       +
               ARGLLST,              Input: Argument length list +
               ARGSLST,              Input: Argument address list+
               ENVCNT,               Input: Environment count    +
               ENVLENS,              Input: Environment length list+
               ENVPARMS,             Input: Environment address list+
               EXITRTNA,             Input: Exit routine address or 0+
               EXITPLA,              Input: Exit Parm list address or 0+
               RETVAL,               Return value: -1 or not return+
               RETCODE,              Return code                 +
               RSNCODE),             Reason code                 +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The caller does not have appropriate permissions to run the specified file. It may lack permission to search a directory named in the *pathname* parameter; or it may lack execute permission for the file to be run; or the file to be run is not a regular file, and the system cannot run files of its type. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRExecNotRegFile.
EAGAIN	Resources were temporarily unavailable.
ECMSERR	An internal error occurred. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRNoStorage.
ELOOP	A loop exists in symbolic links encountered during resolution of the *pathname* argument. This error is issued if more than 8 symbolic links are detected in the resolution of the path name.
EMFILE	The process has reached the maximum number of file descriptors it can have open.
ENAMETOOLONG	The path name is longer than 1023 characters, or some component of the path name is longer than 255 characters. CMS does not support name truncation.
ENFILE	CMS has reached the maximum number of file descriptors it can have open.
ENOENT	No file named *pathname* was found, or no path name was specified. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRExecNmLenZero, JRFileNotThere, JRLinkNotFound, JRNoFileNoCreatFlag, and JRQuiescing.
ENOEXEC	The specified file has execute permission, but is not in the proper format to be a process image file.
ENOMEM	The new process requires more memory than is permitted by the hardware or the operating system. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRExecFileTooBig.
ENOTDIR	Some component of the path name is not a directory.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “alarm (BPX1ALR) — Set an Alarm” on page 18
- “chmod (BPX1CHM) — Change the Mode of a File or Directory by Path Name” on page 28
- “create_external_link (BPX1ELN) — Create a CMS External Link” on page 60
- “fcntl (BPX1FCT) — Control Open File Descriptors” on page 88
- “fork (BPX1FRK) — Create a New Process” on page 96

- “sigpending (BPX1SIP) — Examine Pending Signals” on page 319
- “sigprocmask (BPX1SPM) — Examine or Change a Thread's Signal Mask” on page 321
- “spawn (BPX1SPN) — Spawn a Process” on page 333
- “stat (BPX1STA) -- Get Status Information about a File by Path Name” on page 340
- “umask (BPX1UMK) — Set or Return the File Mode Creation Mask” on page 374.

_exit (BPX1EXI) — End a Process and Bypass the Cleanup

BPX1EXI

status_field

Purpose

Use the _exit (BPX1EXI) service to end the calling process, with the specified status being reported to its parent.

Parameters

status_field

(input,INT,4) is a variable for specifying the status of the ending process. If the contents of this parameter conform to the allowable exit status values, the service provides the contents to the parent when the wait (BPX1WAT) service is called. For a mapping of the this parameter and a description of the conforming status values, see "BPXYWAST — Map the Wait Status Word" on page 486.

Usage Notes

1. If the parent of the ending process has issued a wait call and is waiting for the ending process to end, the status is returned to the parent at once.

   If the parent of the ending process is not waiting, the status is saved. It is returned to the parent if the parent later issues a wait call for the now-ended child.

   If the parent of the ending process does not later wait for the ending process, the ending process's ID (PID) remains in use until the parent ends. Because the number of process IDs is a limited system resource, user and system availability for process IDs may be affected.
2. If the ending process is a session leader, the controlling terminal is disassociated from the session. The controlling terminal can then be acquired by a new controlling process.
3. Child processes of a process that ends are assigned the parent process ID of the init process (whose process ID is 1). The status of these child processes are reported to the init process that frees the PID and system resources associated with the ending process.
4. A **SIGCHLD** signal is sent to the parent of the ending process.
5. Ending a process does not end its child processes directly, however; under the following circumstances a **SIGHUP** signal is sent to a child process that can cause a child process to end:
   - If the ending process is a controlling process, a **SIGHUP** signal is sent to each process in the foreground process group of the controlling terminal belonging to the caller.
   - If ending a process leaves a process group orphaned and any member of that process group is stopped, each member of the process group is sent a **SIGHUP** signal followed by a **SIGCONT** signal.
6. The _exit service does not return to the caller. If it cannot complete its processing successfully, the caller receives an abend.

Characteristics and Restrictions

If the _exit (BPX1EXI) service is invoked with a normal exit status completion code, a normal return to the operating system results.

For a detailed description of the conforming exit status values, see "BPXYWAST — Map the Wait Status Word" on page 486.

Example

The following code ends the program and returns an exit code of 44 to the waiting parent process. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         XC    WAST(WAST#LENGTH),WAST
         MVI   WASTEXITCODE,44       User defined exit code
         SPACE
         CALL  BPX1EXI,              End a process                    +
               (WAST),               Input: Status field              +
               VL,MF=(E,PLIST)       ---------------------------------
```

Related Services

Other callable services related to this service are:

- “close (BPX1CLO) — Close a File or Socket” on page 34
- “cmsprocclp (BPX1MPC) — Clean Up Kernel Resources” on page 38
- “wait (BPX1WAT) — Wait for a Child Process to End” on page 385.

Note: The _exit (BPX1EXI) service is not related to the `exit` shell command and is different from the `exit()` ANSI C routine.

fchaudit (BPX1FCA) — Change Audit Flags for a File by Descriptor

```
BPX1FCA
        file_descriptor
        audit_flags
        option_code
        return_value
        return_code
        reason_code
```

Purpose

Use the fchaudit (BPX1FCA) service to change the types of access to a file to be audited for the security product. You identify the file by its file descriptor.

For the corresponding service using a path name, see "chaudit (BPX1CHA) — Change Audit Flags for a File by Path Name" on page 23.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the file descriptor of the file to be changed.

audit_flags
(input,INT,4) is a variable for specifying the access to be audited. This parameter is mapped by the BPXYAUDT macro. See "BPXYAUDT — Map Flag Values for the chaudit and fchaudit Services" on page 413. Values for this parameter include any combination of the following:

Value	Description
AUDTREADFAIL	Audit failing read requests.
AUDTREADSUCCESS	Audit successful read requests.
ƀ	ƀ
AUDTWRITEFAIL	Audit failing write requests.
AUDTWRITESUCCESS	Audit successful write requests.
ƀ	ƀ
AUDTEXECFAIL	Audit failing execute or search requests.
AUDTEXECSUCCESS	Audit successful execute or search requests.

option_code
(input,INT,4) is a variable for specifying whether you are changing the auditing for the user or for the security auditor. This variable can have the following values:

Value
Meaning

0
The user's auditing is being changed.

1
The security auditor's auditing is being changed.

return_value
(output,INT,4) is a variable where the service returns 0 if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. Audit flags are stored with every object in the Byte File System. They are intended for use by an External Security Manager (ESM) and are not used by native BFS server security or auditing functions. You can use the fchaudit (BPX1FCA) service to change any of the audit flags, even when there is no ESM installed. However, because native BFS does not use the audit flags, they have no effect on security or auditing if no ESM is installed.
2. When no ESM is installed, the authority required to use this service is defined as follows:
   - To change the user audit flags, the user must be either a superuser or the owner of the file.
   - To change the auditor audit flags, the user must be a superuser.
3. When an ESM is installed, the authority requirements to use this service are defined by the ESM. For example, the ESM could define a level of authority called auditor authority, and further declare that auditor authority is required to change the auditor audit flags.

Example

The following code changes the audit for the standard input file to ReadFail, WriteFail and ExecFail. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYAUDT — Map Flag Values for the chaudit and fchaudit Services” on page 413.

```
         MVI   AUDTREADACCESS,AUDTREADFAIL
         MVI   AUDTWRITEACCESS,AUDTWRITEFAIL
         MVI   AUDTEXECACCESS,AUDTEXECFAIL
         MVI   AUDTRSRV,X'00'
         SPACE ,
         CALL  BPX1FCA,              Change audit                      +
               (=A(STDIN_FILENO),    Input: File descriptor            +
               AUDT,                 Input: Audit flags, BPXYAUDT      +
               =A(0),                Input: 0 user, 1 security auditor +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	The *file_descriptor* parameter is not a valid file descriptor.
EINVAL	The *option_code* parameter is incorrect, or the file descriptor refers to an unnamed pipe and this service is not allowed on such a file.
EPERM	The effective user ID of the calling process does not match the owner of the file; or the calling process does not have appropriate privileges; or, if *option_code* indicated that the auditor audit flags were to be changed, then the user may not have had auditor authority.

Return Code	Explanation
EROFS	The specified file is on a read-only file system. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRReadOnlyFS.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “chaudit (BPX1CHA) — Change Audit Flags for a File by Path Name” on page 23
- “stat (BPX1STA) -- Get Status Information about a File by Path Name” on page 340.

fchmod (BPX1FCM) — Change the Mode of a File or Directory by Descriptor

BPX1FCM

file_descriptor
mode
return_value
return_code
reason_code

Purpose

Use the fchmod (BPX1FCM) service to modify the permission bits that control the owner access, group access, and general access to the file. You can use this service to set flags that modify the user ID (UID) and group ID (GID) of the file when it is executed. You can also use this service to set the sticky bit to indicate from where the file should be fetched. You identify the file by its file descriptor.

For the corresponding service using a path name, see "chmod (BPX1CHM) — Change the Mode of a File or Directory by Path Name" on page 28.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the file descriptor of the file whose mode you want to change.

mode
(input,INT,4) is a variable for specifying the new mode of the file. This parameter, which is mapped by the BPXYMODE macro, identifies the file type and the permissions you grant to yourself, to your group, and to any user. See "BPXYMODE — Map Mode Constants" on page 437 for the parameter options.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. File descriptors open at the time of the call to the fchmod (BPX1FCM) service retain the access permission they had at the time the file was opened.
2. For mode bits to be changed, the effective UID of the calling process must match the file's owner UID, or the process must have appropriate privileges.
3. When the mode is changed successfully, the file's change time is updated as well.
4. Setting the set-group-ID-on-execution permission means that when this file is run, through the exec call, the effective GID of the process is set to the file's owner GID, so that the process seems to be running under the GID of the file, rather than that of the actual invoker.

   The set-group-ID-on-execution permission is suppressed (the bit is turned off) if both of the following are true:

- The calling process does not have appropriate privileges.
- The file's owner GID does not match the effective GID or one of the supplementary GIDs of the calling process.

5. Setting the set-user-ID-on-execution permission means that when this file is run the process's effective UID will be set to the file's owner UID, so that the process seems to be running under the UID of the file's owner, rather than that of the actual invoker.

Example

The following code changes the permissions for the standard input file. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structure, see "BPXYMODE — Map Mode Constants" on page 437 and "BPXYFTYP — Map File Type Definitions" on page 423.

```
         XC    S_MODE,S_MODE
         MVI   S_MODE2,S_IRUSR       All permissions
         MVI   S_MODE3,S_IRWXU2+S_IRWXG+S_IRWXO
         SPACE ,
         CALL  BPX1FCM,              Change file modes                +
               (=A(STDIN_FILENO),    Input: File descriptor           +
               S_MODE,               Input: Mode, BPXYMODE, BPXYFTYP  +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	The *file_descriptor* parameter is not a valid file descriptor.
EPERM	The effective UID of the calling process does not match the owner of the file, and the calling process does not have appropriate privileges.
EROFS	The specified file is on a read-only file system. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRReadOnlyFS

For a complete list of return codes for OpenExtensions callable services, see Appendix A, "Return Codes," on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, "Reason Codes," on page 495.

Related Services

Other callable services related to this service are:

- "chmod (BPX1CHM) — Change the Mode of a File or Directory by Path Name" on page 28
- "chown (BPX1CHO) — Change the Owner or Group of a File or Directory" on page 31
- "mkdir (BPX1MKD) — Make a Directory" on page 160
- "open (BPX1OPN) — Open a File" on page 181
- "stat (BPX1STA) -- Get Status Information about a File by Path Name" on page 340.

fchown (BPX1FCO) — Change the Owner and Group of a File or Directory by Descriptor

BPX1FCO

file_descriptor
owner_UID
group_ID
return_value
return_code
reason_code

Purpose

Use the fchown (BPX1FCO) service to change the owner, group, or both of a file. You identify the file by its file descriptor.

For the corresponding service using a path name, see "chown (BPX1CHO) — Change the Owner or Group of a File or Directory" on page 31.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the file descriptor of the file for which you wish to change the owner, group, or both.

owner_UID
(input,INT,4) is a variable for specifying the new owner UID assigned to the file, or the present value if there is no change. This parameter must be specified.

group_ID
(input,INT,4) is a variable for specifying the new group ID assigned to the file, or the present value if there is no change. This parameter must be specified.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. The fchown (BPX1FCO) service changes the owner UID and group ID of a file. Only a process with superuser authority can change the owner UID of a file.
2. The group ID of a file can be changed by a process if the process has appropriate privileges, or if a process meets all of the following conditions:
   - The effective UID of the process matches the file's owner UID.
   - The *owner_UID* value specified in the change request also matches the file's owner UID.

- The *group_ID* value specified in the change request is the effective GID,or one of the supplementary GIDs, of the calling process.

3. The set-user-ID-on-execution and set-group-ID-on-execution permissions of the file mode are automatically turned off.
4. If the change request is successful, the change time for the file is updated.
5. Values for both *owner_UID* and *group_ID* must be specified as they are to be set. Ifit is desired to change only one of these values, the other must be set to its present value to remain unchanged.

Example

The following code changes the owner and group for the standard input file. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   GROUPID,..            Group ID
         MVC   USERID,..             User ID
         SPACE ,
         CALL  BPX1FCO,              Change the owner and group of file+
               (=A(STDIN_FILENO),    Input: File descriptor            +
               USERID,               Input: New user ID for file       +
               GROUPID,              Input: New group ID for file      +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	The *file_descriptor* parameter is not a valid file descriptor.
EINVAL	The *owner_UID* or *group_ID* parameter is incorrect, or *file_descriptor* refers to an unnamed pipe and this service is not allowed on such a file.
EPERM	The calling process does not have appropriate privileges.
EROFS	The specified file is on a read-only file system. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRReadOnlyFS.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “chown (BPX1CHO) — Change the Owner or Group of a File or Directory” on page 31
- “fchmod (BPX1FCM) — Change the Mode of a File or Directory by Descriptor” on page 84
- “fstat (BPX1FST) -- Get Status Information about a File by Descriptor” on page 102.

fcntl (BPX1FCT) — Control Open File Descriptors

BPX1FCT

file_descriptor
action
argument
return_value
return_code
reason_code

Purpose

Use the fcntl (BPX1FCT) service to perform general control functions for open files and sockets. This service retrieves or sets the file descriptor flags, file status flags, and locking information.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the descriptor for a file or socket.

This parameter must specify an opened file descriptor, except when the *action* parameter is F_CLOSFD. In that case, this file descriptor is not expected to be in use.

action
(input,INT,4) is a variable for specifying the action to be performed. This parameter is mapped by the BPXYFCTL macro. For a list of action values, see "BPXYFCTL — Map Command Values and Flags for the fcntl Service" on page 422.

argument
(input/output,INT,4) is a variable for specifying either an argument or zero. The type of argument you can use depends upon the action requested in the previous parameter:

Action	Argument
F_CLOSFD	*file_descriptor_2*
F_DUPFD	*file_descriptor_2*
F_DUPFD2	*file_descriptor_2*
F_GETFD	0
F_GETFL	0
F_GETLK	*lock_information*
F_SETFD	*file_descriptor_flags*
F_SETFL	*file_status_flags*
F_SETLK	*lock_information*
F_SETLKW	*lock_information*

Argument Descriptions:

file_descriptor_2
is the name of a fullword containing a file descriptor. The function performed by the service depends on the specified action:

Action
Function Performed

F_DUPFD
The service returns the lowest file descriptor equal to or greater than *file_descriptor_2* not already associated with an open file.

F_DUPFD2
The service returns a file descriptor equal to *file_descriptor_2*. If the file identified by *file_descriptor_2* is already in use, the file is closed and *file_descriptor* is duplicated. If *file_descriptor* is equal to *file_descriptor_2*, *file_descriptor_2* is returned without closing it.

F_CLOSFD
The *file_descriptor_2* argument specifies the upper limit for the range of file descriptors to be closed, and *file_descriptor* specifies the lower limit. If a -1 is specified for *file_descriptor_2*, all file descriptors greater than or equal to the lower limit are closed.

file_descriptor_flags
is the name of a fullword containing the file descriptor flags to be set or retrieved for *file_descriptor*.

To get *file_descriptor_flags*, specify action F_GETFD. If successful, *return_value* maps to the bit settings of *file_descriptor_flags*.

Similarly, to set *file_descriptor_flags*, specify action F_SETFD and use the mapping to set or reset *file_descriptor_flags* to the desired value.

Note: After the FCTLCLOFORK flag has been set on, it cannot be set off again.

File descriptor flags are mapped by the BPXYFCTL macro. See "BPXYFCTL — Map Command Values and Flags for the fcntl Service" on page 422.

file_status_flags
is the name of a fullword containing the file status flags to be set or retrieved for *file_descriptor*.

To get *file_status_flags*, specify action F_GETFL. If successful, *return_value* maps to the bit settings of *file_status_flags*.

Similarly, to set *file_status_flags*, specify action F_SETFL and use the mapping to set or reset *file_status_flags* to the desired value. Only the O_APPEND, O_NONBLOCK, and O_SYNC flags are set when the action is F_SETFL; any other flags specified are ignored.

File status flags are used to set some of the open flags that are mapped by the BPXYOPNF macro. For the mapping of the file status flags, see "BPXYOPNF — Map Flag Values for the open and fcntl Services" on page 447.

Two masks are available for use with the return value from an F_GETFL request. You can extract the file access mode flags from the return value using the O_ACCMODE mask, or use the O_GETFL mask to extract both the file access mode and file status flags.

lock_information
is the name of a fullword containing a pointer to a structure containing information on a file segment for which locks are to be set, cleared, or queried.

The *lock_information* argument is mapped by the BPXYBRLK macro as follows:

Word	Description
0	L_TYPE field: Bytes 0–1 specify the type of lock being set, cleared, or queried.
0	L_WHENCE field: Bytes 2–3 specify how the lock offset is to be determined.
1–2	L_START field specifies the starting byte offset of the lock to be set, cleared, or queried. This is a doubleword value.
3–4	L_LEN field specifies the length of the byte range to be set, cleared, or queried. This is a doubleword value.

Word	Description
5	L_PID field, upon return from a F_GETLK request, contains the process ID of the process holding the blocking lock, provided that one was found.

See “BPXYBRLK — Map the Byte Range Lock Request for the fcntl Service” on page 414. For more information about these fields, see "File Locking" in “Usage Notes” on page 90.

return_value
(output,INT,4) is a variable where, if the request is successful, the service returns one of the following values, according to the specified action and argument. If the request is not successful, a -1 is returned.

Action	Argument	Return Value
F_CLOSFD	*file_descriptor_2*	0
F_DUPFD	*file_descriptor_2*	*file_descriptor*
F_DUPFD2	*file_descriptor_2*	*file_descriptor*
F_GETFD	0	*file_descriptor_flags*
F_GETFL	0	*file_status_flags*
F_GETLK	*lock_information*	*lock_information*
F_SETFD	*file_descriptor_flags*	0
F_SETFL	*file_status_flags*	0
F_SETLK	*lock_information*	0
F_SETLKW	*lock_information*	0

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

Closing Files: A process can use the BPX1FCT service to close a range of file descriptors. The *file_descriptor_2* argument must be greater than or equal to *file_descriptor*, or it can also be -1, which indicates that all file descriptors greater than or equal to *file_descriptor* are to be closed.

Use of F_CLOSFD is meant to be consistent with the close service. You cannot close file descriptors that could not also be closed using the close service, BPX1CLO.

If a file descriptor cannot be closed, it is considered an error, but the request continues with the next file descriptor in the range. File descriptors that are not in use are ignored.

File Locking: A process can use the fcntl (BPX1FCT) service to lock out other cooperating processes from part of a file, so that the process can read or write to that part of the file without interference from others. This can ensure data integrity when several processes have a file accessed concurrently.

Requests to lock files in NFS-mounted directories are ignored.

Locking operations are controlled with a structure mapped by BPXYBRLK, whose format is described under the *lock_information* parameter. This structure is needed whether the request be for setting a lock,

releasing a lock, or querying a particular byte range for a lock. The following is a more detailed description of the BPXYBRLK structure.

L_TYPE Field — This field specifies the type of lock to be set, cleared, or queried. Valid values for L_TYPE are as follows:

Value
Description

F_RDLCK
Indicates a *read lock*. Specified as a halfword integer value of 1, this is also known as a *shared lock*. This type of lock specifies that the process can read the locked part of the file, and other processes cannot write on that part of the file in the meantime. A process can change a held write lock, or any part of it, to a read lock, thereby making it available for other processes to read. Multiple processes can have read locks on the same part of a file simultaneously. To establish a read lock, a process must have the file accessed for reading.

F_WRLCK
Indicates a *write lock*. Specified as a halfword integer value of 2, this is also know as an *exclusive lock*. This type of lock indicates that the process can write on the locked part of the file, without interference from other processes. If one process puts a write lock on part of a file, no other process can establish a read lock or write lock on that same part of the file. A process cannot put a write lock on part of a file if there is already a read lock on an overlapping part of the file, unless that process is the only owner of that overlapping read lock. In such a case, the read lock on the overlapping section is replaced by the write lock being requested. To establish a write lock, a process must have the file accessed for writing.

F_UNLCK
Indicates unlock. Specified as a halfword integer value of 3, this is used to unlock all locks held on the given range by the requesting process.

L_WHENCE Field — This field specifies how the byte range offset is to be found within the file. The use of this field for the fcntl (BPX1FCT) service parallels its processing for the lseek (BPX1LSK) service. See "lseek (BPX1LSK) — Change the File Offset" on page 154 for more information.

Valid values for L_WHENCE are as follows:

Value
Description

SEEK_SET
Indicates the start of the file. It is specified as a halfword integer value of 0.

SEEK_CUR
Indicates the current file offset in the file. It is specified as a halfword integer value of 1.

SEEK_END
Indicates the end of the file. It is specified as a halfword integer value of 2.

L_START Field — This field identifies the part of the file to be locked, unlocked, or queried. The part of the file affected by the lock begins at this offset from the location specified by the L_WHENCE field. For example, if L_WHENCE is SEEK_CUR and L_START is 10, a F_SETLK request attempts to set a lock beginning 10 bytes past the current cursor position. The L_START value may be negative, provided that when added to the offset indicated by the L_WHENCE position, the resulting offset does not extend beyond the beginning of the file.

Note: Though you cannot request a byte range that begins or extends beyond the beginning of the file, you can request a byte range that starts or extends beyond the end of the file.

The use of the L_START field for the fcntl (BPX1FCT) service parallels its processing for the lseek (BPX1LSK) service. See "lseek (BPX1LSK) — Change the File Offset" on page 154 for more information.

L_LEN Field — This field gives the size of the locked part of the file, in bytes. The area affected begins at L_START and ends at L_START+L_LEN-1. The value specified for L_LEN cannot be negative. If a negative value is specified for L_LEN, a return value of -1 and a return code of EINVAL are returned. If L_LEN is

zero, the locked part of the file begins at the position specified by L_WHENCE and L_START, and extends to the end of the file.

L_PID Field — This field identifies the process ID of the process that holds the lock found on an F_GETLK request, if one was found.

Obtaining Locks: Locks can be set by specifying F_SETLK as the action parameter for the fcntl (BPX1FCT) service. If the lock cannot be obtained, a *return_value* of -1 is returned along with an appropriate *return_code* and *reason_code*. You can also use F_SETLK to release locks already held, by setting L_TYPE to F_UNLCK.

You can also set locks by specifying F_SETLKW as the action parameter. If the lock cannot be obtained because another process has a lock on all or part of the requested range, the F_SETLKW request waits until the specified range becomes free and the request can be completed. You can also use F_SETLKW to release locks already held, by setting L_TYPE to F_UNLCK.

If a signal interrupts a call to the fcntl (BPX1FCT) service while it is waiting in a F_SETLKW operation, the function returns with a *return_value* of -1, and a *return_code* of EINTR.

F_SETLKW operations have the potential for encountering deadlocks. This happens when process A is waiting for process B to unlock a region, and B is waiting for A to unlock a different region. If the system detects that a F_SETLKW might cause a deadlock, the fcntl (BPX1FCT) service returns with a *return_value* of -1 and a *return_code* of EDEADLK.

Determining Lock Status: A process can determine locking information about a file using F_GETLK as the action parameter for the fcntl (BPX1FCT) service. In this case, the *argument* parameter should specify a pointer to a structure mapped by the BPXYBRLK macro. This structure should describe a lock operation that the caller would like to perform. When the service returns, the structure is modified to describe the first lock found that would prevent the proposed lock operation from completing successfully.

If a lock is found that would prevent the proposed lock from being set, the F_GETLK request returns a modified structure whose:

- L_WHENCE value is always SEEK_SET
- L_START value gives the offset of the locked portion from the beginning of the file
- L_LEN value is set to the length of the locked portion of the file
- L_PID value is set to the process ID of the process that is holding the lock.

If there are no locks that would prevent the proposed lock operation from completing successfully, the returned structure is modified to have an L_TYPE of F_UNLCK, but otherwise remains unchanged.

Multiple Lock Requests: A process can have several locks on a file simultaneously, but can have only one type of lock set on any given byte. Therefore, if a process puts a new lock on part of a file that it had previously locked, the process has only one lock on that part of the file and the lock type is the one given by the most recent locking operation.

Releasing Locks: When an F_SETLK or F_SETLKW request is made to unlock a byte region of a file, all locks held by that process within the specified region are released. In other words, each byte specified on an unlock request is freed from any lock that is held against it by the requesting process.

All of a process's locks on a file are removed when the process closes a file descriptor for that file. Locks are not inherited by a child process created with the spawn (BPX1SPN) service. For more information, see “spawn (BPX1SPN) — Spawn a Process” on page 333.

Important Note:
All locks are advisory only. Processes can use locks to inform each other that they want to protect parts of a file, but locks don't prevent I/O on the locked parts. A process that has appropriate permissions on a file can perform whatever I/O it chooses, regardless of what locks are set. Therefore, file locking is only a convention, and it works only when all processes respect the convention.

Example

The code for the first example duplicates the standard error file descriptor to a file descriptor greater than or equal to FILEDES2. The code for the second example set a shared byte range lock. These examples follow the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYFCTL — Map Command Values and Flags for the fcntl Service” on page 422, “BPXYBRLK — Map the Byte Range Lock Request for the fcntl Service” on page 414, and “BPXYOPNF — Map Flag Values for the open and fcntl Services” on page 447.

```
* for 2nd parm F_DUPFD, F_DUPFD2           3rd parm file desc no..
* for 2nd parm F_GETFD, F_GETFL            3rd parm 0
* for 2nd parm F_SETFD                     3rd parm BPXYFCTL
* for 2nd parm F_GETLK, F_SETLK, F_SETLKW  3rd parm BPXYBRLK
* for 2nd parm F_SETFL                     3rd parm BPXYOPNF
         SPACE ,
* Example 1 - duplicate file descriptor
         MVC   FILEDES2,=F'20'       Get free file descriptor >= 20
         SPACE ,
         CALL  BPX1FCT,              General purpose file control      +
               (=A(STDERR_FILENO),   Input: File descriptor            +
               =A(F_DUPFD),          Input: Action, BPXYFCTL           +
               FILEDES2,             Input: Argument #/0/FCTL/BRLK/OPNF+
               RETVAL,               Return value: 0, -1 or action     +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
         SPACE ,
* Example 2 - duplicate file descriptor
         MVC   FILEDES2,=F'20'       Get next higher file descriptor
         LA    R15,BRLK
         ST    R15,BRLKA
         XC    BRLK(BRLK#LENGTH),BRLK      Null out BRLK
         MVC   L_TYPE,=AL2(F_RDLCK)        Lock type = shared
         MVC   L_WHENCE,=AL2(SEEK_CUR)     Whence = from current cursor
         SPACE ,
         CALL  BPX1FCT,              General purpose file control      +
               (=A(STDERR_FILENO),   Input: File descriptor            +
               =A(F_SETLK),          Input: Action, BPXYFCTL           +
               BRLKA,                Input: Argument #/0/FCTL/BRLK/OPNF+
               RETVAL,               Return value: 0, -1 or action     +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The calling process asked to set a lock, but the lock conflicts with a lock on an overlapping part of the file already set by another process.

Return Code	Explanation
EBADF	The request was not accepted, for one of these reasons: • The *file_descriptor* parameter does not specify a valid, open file descriptor. • The request was to set a read lock, but the file is open for writing only. • The request was to set a write lock, but the file is open for reading only. • The *file_descriptor* was opened with an opendir request. Many of the other requests are rejected for an opendir filedes. • If the action requested was F_DUPFD2, this error indicates that *file_descriptor_2* was negative, or was equal to or greater than the highest file descriptor value allowed for the process. The following reason codes can accompany this return code: JRFdTooBig, JRNegFileDes.
EDEADLK	The action requested was F_SETLKW; the potential for deadlock was detected.
EINTR	The service was interrupted by a signal while processing a F_SETLKW request.
EINVAL	The request was not accepted, for one of these reasons: • If the action requested was F_DUPFD, this error indicates that *file_descriptor_2* was negative, or was equal to or greater than the highest file descriptor value allowed for the process. • If the action requested was F_SETLK or F_SETLKW, the file specified by *file_descriptor* does not support locking, or the *lock_information* parameter contains incorrect values. • The action requested was F_CLOSFD and the file descriptor specified by *file_descriptor_2* was less than *file_descriptor*, but was not equal to -1. • An incorrect action was requested. The following reason codes can accompany this return code: JRBrlmBadFileType, JRBrlmBadL_Type, JRBrlmBadL_Whence, JRBrlmInvalidRange, JRFdTooBig, JRFd2TooSmall, JRNegFileDes.
EMFILE	The action requested was F_DUPFD. The process has already reached its maximum number of file descriptors, or there is no file descriptor available greater than *file_descriptor_2*.
ENOTSOCK	*file_descriptor* does not refer to a valid socket descriptor. The following reason code can accompany this return code: JRMustBeSocket.
EPERM	The action requested was F_CLOSFD, and at least one of the file descriptors in the specified range remains open. For a description of the file descriptors that cannot be closed with F_CLOSFD, see the usage notes.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “close (BPX1CLO) — Close a File or Socket” on page 34
- “exec (BPX1EXC) — Run a Program” on page 72
- “fork (BPX1FRK) — Create a New Process” on page 96

- “lseek (BPX1LSK) — Change the File Offset” on page 154
- “open (BPX1OPN) — Open a File” on page 181
- “spawn (BPX1SPN) — Spawn a Process” on page 333.

fork (BPX1FRK) — Create a New Process

BPX1FRK

process_ID
return_code
reason_code

Purpose

Use the fork (BPX1FRK) service to create a new process. The new process (known as the *child process*) is a duplicate of the process that calls fork (known as the *parent process*).

Note: The OpenExtensions implementation of the fork (BPX1FRK) service has some limitations not found in other implementations (see "Characteristics and Restrictions" on page 97). In certain situations, you may need to modify your application to accommodate these limitations. To avoid the limitations of fork (BPX1FRK), you should consider modifying your application to use spawn (BPX1SPN). For information about converting `fork()` usage in a C/C++ program to `spawn()`, see the *z/VM: CMS Application Development Guide*.

Parameters

process_ID
(output,INT,4) is a variable where the service returns a process ID, `0`, or `-1`:

- Upon successful completion, fork returns the process ID of the newly-created child to the calling (parent) process.
- Because the child is a duplicate, it contains the same call to fork that was in the parent. Execution of the child process begins with the fork call in the child returning a *process_ID* of `0`. The child then proceeds with normal execution.
- If *process_ID* is returned as `-1` to the calling process, the fork request was not successful, and no child process was created.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *process_ID* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *process_ID* is `-1`.

Usage Notes

1. Although the child process is a duplicate of the parent process, there are the following differences:
   - The child has a unique process ID (PID) that does not match any active process group ID.
   - The child has a different parent process ID (namely, the process ID of the process that called fork).
   - The child has its own copy of the parent's file descriptors. Each file descriptor in the child refers to the same open file as the corresponding file descriptor in the parent.
   - The child has its own copy of the parent's open directory streams. Each open directory stream in the child can share directory stream positioning with the corresponding directory stream of the parent.
   - The process and system utilization times for the child are set to zero.
   - Any file locks previously set by the parent are not inherited by the child.

- The child process has no interval timers set (similar to the results of a call to the alarm service with the *seconds* parameter specified as zero).
- The child has no pending signals.

In other respects, the child is identical to the parent.

2. The child process inherits all shared memory attachments attached to the calling process. The internal values of the number of processes attached to each shared memory segment (SHM_NATTCH field of the SHMID_DS data structure in the BPXYSHM macro) will be incremented.

Characteristics and Restrictions

1. You must issue the CMS command OPENVM SET FORK ON before running an application that uses the `fork()` function. If the CMS FORK flag is not turned on, the application will receive a return value of `-1`, a return code of ENOSYS, and a reason code of JRFunctNotSupported.
2. You must run the application as a POSIX(ON) application. If this flag is not turned on, the application will receive a return value of `-1`, a return code of ENOSYS, and a reason code of JRFunctNotSupported.
3. The child process is not allowed to issue an `exit()` call or to call any function that will invoke `exit()` before the child process issues the `exec()` function. Any attempt to exit the child process before the `exec()` is issued will result in a X'AE5' abnormal end code.
4. The child process is not allowed to issue any function that will cause the child process to be blocked (for example, a pipe `read()` or a `pause()`), before the child issues the `exec()` function. Any attempt to exit the child process before the `exec()` is issued will result in a X'AE6' abnormal end code.
5. Any local variables in the application that are changed in the child process before the `exec()` is issued will be changed in the parent process as well. This is because the child and parent processes are still using the same program storage. The `exec()` function causes the child process to begin using its own program storage.
6. Any global or environment variables in the application that are changed in the child process before the `exec()` is issued will be changed in the parent process as well. This is because the child and parent processes are still using the same program storage. The `exec()` function causes the child process to begin using its own program storage.

Example

The following code creates a new process. The next sequential instruction gets control from both the parent process (RETVAL=child process ID) and from the child process (RETVAL=0). If RETVAL=`-1`, the fork failed.

```
         CALL  BPX1FRK,              Create a new process (fork)       +
               (RETVAL,              Return value: -1, 0, child's PID  +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EAGAIN	One of the following conditions is true: • You have already reached the maximum number of processes you are allowed to run. The following reason code can accompany this return code: JRMaxProc.

Return Code	Explanation
ENOSYS	The CMS OPENVM FORK option is not enabled. Issue the OPENVM SET FORK ON command to turn on the CMS OPENVM FORK option. The following reason code can accompany this return code: JRFuncNotSupported.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “exec (BPX1EXC) — Run a Program” on page 72
- “spawn (BPX1SPN) — Spawn a Process” on page 333

fpathconf (BPX1FPC) — Determine Configurable Path Name Variables Using a Descriptor

BPX1FPC

file_descriptor
name
return_value
return_code
reason_code

Purpose

Use the fpathconf (BPX1FPC) service to determine the current value of a configurable limit or option (variable) associated with a file or directory identified by its file descriptor.

For the corresponding service using a path name, see "pathconf (BPX1PCF) — Determine Configurable Path Name Variables Using Path Name" on page 194.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the file descriptor of the file.

name
(input,INT,4) is a variable for specifying the path name variable to be returned. Use the BPXYPCF macro. See "BPXYPCF — Map Command Values for the pathconf and fpathconf Services" on page 448. The path name variables you can specify are:

Path Name Variable	Description
PC_CHOWN_RESTRICTED	The change ownership service, chown (BPX1CHO), is restricted to a process with appropriate privileges, and to changing the group ID (GID) of a file to only the effective group ID of the process or one of its supplementary group IDs.
PC_LINK_MAX	Maximum value of a file's link count.
PC_MAX_CANON	Maximum number of bytes in a terminal canonical input line.
PC_MAX_INPUT	Minimum number of bytes for which space will be available in a terminal input queue; therefore, the maximum number of bytes a portable application may require to be typed as input before reading them.
PC_NAME_MAX	Maximum number of bytes in a file name (not a string length; count excludes a terminating null).
PC_NO_TRUNC	Path name components longer than 255 bytes generate an error.
PC_PATH_MAX	Maximum number of bytes in a path name (not a string length; count excludes a terminating null).
PC_PIPE_BUF	Maximum number of bytes that can be written atomically when writing to a pipe.

Path Name Variable	Description
PC_VDISABLE	Terminal special characters maintained by the system can be disabled using this character value. For information on querying and setting these special characters, see "tcgetattr (BPX1TGA) — Get the Attributes for a Terminal" on page 358 or "tcsetattr (BPX1TSA) — Set the Attributes for a Terminal" on page 365.

return_value
(output,INT,4) is a variable where the service returns the current value of the path name variable specified in the *name* parameter, or -1 if the request is not successful.

If the path name variable is PC_CHOWN_RESTRICTED and this option is active, the return value is set to 1. If this option is not active, the return value is set to 0.

If the path name variable is PC_NO_TRUNC and this option is active, the return value is set to 1. If this option is not active, the return value is set to 0.

If the path name variable does not have a limit for the specified file, the return value is set to -1 and the return code and reason code remain unchanged.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1. If the path name variable does not have a limit for the specified file, the return value is set to -1 and the return code is unchanged.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1. If the path name variable does not have a limit for the specified file, the return value is set to -1 and the reason code is unchanged.

Usage Notes

1. If *name* is PC_MAX_CANON, PC_MAX_INPUT, or PC_VDISABLE, and *file_descriptor* does not refer to a terminal file, the service returns return value -1 and return code EINVAL.
2. If *name* is PC_NAME_MAX, PC_PATH_MAX, or PC_NO_TRUNC, and *file_descriptor* does not refer to a directory, the service still returns the requested information using the parent directory of the specified file.
3. If *name* is PC_PIPE_BUF:
   - If *file_descriptor* refers to a pipe or a FIFO, the return value applies to the referred-to object.
   - If *file_descriptor* refers to a directory, the return value applies to any FIFOs that exist or can be created within the directory.
   - If *file_descriptor* refers to any other type of file, the service returns return value -1 and return code EINVAL.
4. If *name* is to PC_LINK_MAX and *file_descriptor* refers to a directory, the return value applies to the directory.

Example

The following code obtains the configurable option associated with the pipe buffer. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structure, see "BPXYPCF — Map Command Values for the pathconf and fpathconf Services" on page 448.

```
         MVC   FILEDESC,..           From opendir
         SPACE ,
         CALL  BPX1FPC,              Get configurable pathname variable+
               (FILEDESC,            Input: Directory file descriptor  +
               =A(PC_PIPE_BUF),      Input: Configurables      BPXYPCF +
               RETVAL,               Return value: 0, -1 or variable   +
               RETCODE,              Return code                       +
```

```
                RSNCODE),             Reason code                         +
                VL,MF=(E,PLIST)       ------------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	The *file_descriptor* argument is not a valid file descriptor.
EINVAL	Refer to the Usage Notes for situations where this return code is returned.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Another callable service related to this service is:

- “pathconf (BPX1PCF) — Determine Configurable Path Name Variables Using Path Name” on page 194.

fstat (BPX1FST) -- Get Status Information about a File by Descriptor

BPX1FST

file_descriptor
status_area_length
status_area
return_value
return_code
reason_code

Purpose

Use the fstat (BPX1FST) service to obtain status information about a file identified by its file descriptor.

For the corresponding service using a path name, see "stat (BPX1STA) -- Get Status Information about a File by Path Name" on page 340.

To obtain status information about a symbolic link, rather than for a file to which it refers, see "lstat (BPX1LST) — Get Status Information about a File or Symbolic Link by Path Name" on page 157.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the file descriptor for the file.

status_area_length
(input,INT,4) is a variable for specifying the length of the *status_area* parameter. To determine the value of *status_area_length*, use the BPXYSTAT macro. See "BPXYSTAT — Map the File Status Structure for the stat Service" on page 473.

status_area
(output,CHAR,length of BPXYSTAT or *status_area_length*, whichever is less.) is a variable for the area where the service returns the status information for the file. The status area is mapped by the BPXYSTAT macro. See "BPXYSTAT — Map the File Status Structure for the stat Service" on page 473.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. All the modified data in the file specified by *file_descriptor* is written to permanent storage when this service is requested. See "fsync (BPX1FSY) — Write Changes to Direct-Access Storage" on page 106.
2. All time fields in the *status_area* are in POSIX format, which is the number of seconds since January 1, AD 1970, 00:00:00 UTC. If you need to perform conversions on POSIX times, see the DateTimeSubtract CSL routine in the *z/VM: CMS Application Multitasking* or the DATECONVERT stage in the *z/VM: CMS Pipelines User's Guide and Reference*.

3. The file mode field in the status area is mapped by the BPXYMODE macro, and the file type field within the mode area is mapped by the BPXYFTYP macro. See "BPXYMODE — Map Mode Constants" on page 437 and "BPXYFTYP — Map File Type Definitions" on page 423.

Example

The following code gets the file status for the file opened as FILEDESC. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structure, see "BPXYSTAT — Map the File Status Structure for the stat Service" on page 473.

```
         MVC   FILEDESC,..           File descriptor from open
         SPACE ,
         CALL  BPX1FST,              Get file status of file descriptor+
               (FILEDESC,            Input: File descriptor            +
               STATL,                Input: Length of buffer needed    +
               STAT,                 Buffer, mapped by BPXYSTAT        +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	The *file_descriptor* parameter does not identify a known file.
ECMSERR	An internal error occurred.
EINVAL	Parameter error. For example, a zero-length buffer was passed. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRBuffTooSmall.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, "Return Codes," on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, "Reason Codes," on page 495.

Related Services

Other callable services related to this service are:

- "fcntl (BPX1FCT) — Control Open File Descriptors" on page 88
- "open (BPX1OPN) — Open a File" on page 181
- "stat (BPX1STA) -- Get Status Information about a File by Path Name" on page 340.

fstatvfs (BPX1FTV) — Get Status Information about File System by Descriptor

```
BPX1FTV
        file_descriptor
        status_area_length
        status_area
        return_value
        return_code
        reason_code
```

Purpose

Use the fstatvfs (BPX1FTV) service to obtain status information about a file system by its file descriptor.

For the corresponding service using a path name, see "statvfs (BPX1STV) — Get Status Information about a File System by Path Name" on page 343. For the corresponding service using a file system name, see "w_statvfs (BPX1STF) — Get Status Information about a File System by File System Name" on page 407.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the file descriptor for a file.

status_area_length
(input,INT,4) is a variable for specifying the length of the *status_area* parameter.

status_area
(output,CHAR,*status_area_length*) is a variable for the area where the service returns the status information for the file system. This area is mapped by the BPXYSSTF macro. See "BPXYSSTF — Map the File System Status Structure" on page 471.

return_value
(output,INT,4) is a variable where the service returns the length of the data returned in *status_area* if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. If the passed *status_area_length* is not less than or equal to zero, it is not considered an error for the *status_area_length* to be insufficient to hold the requested information. (In other words, future expansion is allowed for.) As much information as can fit is written to *status_area*, and this amount is returned.
2. The amount of valid data returned in the *status_area* is indicated by the *return_value*. This allows for differences in the release levels of OS/390® UNIX and the physical file systems.

Example

The following code requests status information about the target file system. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structure, see "BPXYMTM — Map the Modes for the mount and umount Services" on page 445.

```
MVC   FILEDESC,..          File descriptor from open
  SPACE ,
  CALL  BPX1FTV,             Get file system status           +
        (FILEDESC,           Input: File descriptor           +
        SSTFL,               Input: Length of BPXYSSTF        +
        SSTF,                Buffer, BPXYSSTF                 +
        RETVAL,              Return value: Status length or -1 +
        RETCODE,             Return code                      +
        RSNCODE),            Reason code                      +
        VL,MF=(E,PLIST)      ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EAGAIN	Information is temporarily unavailable. This can occur because the mount process for the file system is incomplete.
EBADF	The *file_descriptor* argument is not a valid file descriptor.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, "Return Codes," on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, "Reason Codes," on page 495.

Related Services

Other callable services related to this service are:

- "statvfs (BPX1STV) — Get Status Information about a File System by Path Name" on page 343.
- "w_statvfs (BPX1STF) — Get Status Information about a File System by File System Name" on page 407.

fsync (BPX1FSY) — Write Changes to Direct-Access Storage

BPX1FSY

file_descriptor
return_value
return_code
reason_code

Purpose

Use the fsync (BPX1FSY) service to make changes to a file permanent by writing the changes on the direct-access storage device that holds the file. You identify the file by its file descriptor.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the file descriptor of the file for which changes are to be written to permanent storage.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return codes is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Note

The fsync (BPX1FSY) service causes all modified data in the specified file to be written to the direct-access storage device that holds the file. On return from a successful call, all updates have been saved on the direct-access storage that holds the file.

Characteristics and Restrictions

The file identified by *file_descriptor* must be open for writing when the fsync (BPX1FSY) service is called.

Example

The following code writes file descriptor changes to permanent storage. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   FILEDESC,..          File descriptor from open
         SPACE ,
         CALL  BPX1FSY,             Write changes to permanent storage+
               (FILEDESC,           Input: File descriptor            +
               RETVAL,              Return value: 0 or -1             +
               RETCODE,             Return code                       +
               RSNCODE),            Reason code                       +
               VL,MF=(E,PLIST)      ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	The *file_descriptor* parameter does not specify a valid, open file.
ECMSERR	An internal error occurred.
EINVAL	The file is not a regular file.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “open (BPX1OPN) — Open a File” on page 181
- “write (BPX1WRT) — Write to a File or Socket” on page 401.

ftruncate (BPX1FTR) — Truncate a File

BPX1FTR

file_descriptor
file_length
return_value
return_code
reason_code

Purpose

Use the ftruncate (BPX1FTR) service to make a file shorter. You identify the file by its file descriptor.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the file descriptor of the file to be truncated.

file_length
(input,INT,8) is a variable for specifying the number of bytes to remain in the file after truncation.

This variable is a doubleword to accommodate large files. For processing with a singleword value, propagate the sign bit through the second word, so the final doubleword value has as valid sign. This service accepts only positive values.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. The ftruncate (BPX1FTR) service truncates the file to *file_length* bytes, beginning at the first byte of the file. All data from *file_length* to the original end of the file is removed.
2. Full blocks are returned to the file system so that they can be used again, and the file size is changed to the lesser of *file_length* or the current length of the file. The file offset is not changed.
3. If the ftruncate (BPX1FTR) service completes successfully, it clears the set-user-ID, set-group-ID, and save-text (sticky bit) attributes of the file unless the caller has authority to access the root.

Characteristics and Restrictions

The file specified must be a regular file, open for writing.

Example

The following code truncates the file described by FILEDESC after 512 bytes. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   FILEDESC,..           File descriptor from open
         MVC   NEWLEN(8),=D'512'
         SPACE ,
         CALL  BPX1FTR,              Truncate a file                  +
               (FILEDESC,            Input: File descriptor           +
               NEWLEN,               Input: Length to keep            +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	The *file_descriptor* parameter does not specify a valid, open file.
EINVAL	The file is not a regular file, or it is opened read-only, or the *file_length* specified is negative. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRTrNegOffset, JRTrNotRegFile, and JRTrOpenedRO.
EROFS	The specified file is on a read-only file system. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRTrMountedRO.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Another callable service related to this service is:

- “open (BPX1OPN) — Open a File” on page 181.

getclientid (BPX1GCL) — Obtain the Calling Program's Identifier

BPX1GCL

function_code
domain
client_ID
return_value
return_code
reason_code

Purpose

Use the getclientid (BPX1GCL) service to obtain the calling program's identifier.

Parameters

function_code
(input,INT,4) is a variable for specifying the function to be performed:

1
Return the caller's name and task identifiers

domain
(input,INT,4) is a variable for specifying a value that represents the communications domain in which the sockets are to be given and taken. This must be AF_INET or AF_INET6. This value is defined in the BPXYSOCK macro. See "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465.

client_ID
(output,CHAR,length of BPXYCID) is a variable where the service returns a structure containing the requested data:

For function code 1, the returned *client_ID* is filled in as follows:

CIdDomain
Input Domain

CIdName
Calling program's VM user ID, left-justified and padded with blanks

CIdTask
Calling program's subtask identifier

CIdReserved
Binary zeros

This field is mapped by the BPXYCID macro. See "BPXYCID — Map the Client ID Structure" on page 415.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Note

The client ID output of getclientid is intended to be used as the input client ID of the givesocket (BPX1GIV) and takesocket (BPX1TAK) services.

Example

The following code obtains the client ID information for caller. This information is used on givesocket (BPX1GIV) and takesocket (BPX1TAK). This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYCID — Map the Client ID Structure” on page 415.

```
         CALL  BPX1GCL,              get clientid information          +
               (=F'2',               Input: Function code of 2         +
               =A(AF_INET),          Input: Domain of AF_INET          +
               CID,                  Output: Clientid information      +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EAFNOSUPPORT	The address family is not supported.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “givesocket (BPX1GIV) — Give a Socket to Another Program” on page 142
- “takesocket (BPX1TAK) — Acquire a Socket from Another Program” on page 350

getcwd (BPX1GCW) — Get the Path Name of the Working Directory

BPX1GCW

buffer_length

buffer

return_value

return_code

reason_code

Purpose

Use the getcwd (BPX1GCW) service to get the path name of the working directory.

Parameters

buffer_length
(input,INT,4) is a variable for specifying the length of the buffer where the service returns the path name of the working directory. The *buffer_length* must be large enough to accommodate the actual length of the path name plus one (for the terminating null).

buffer
(output,CHAR,*buffer_length*) is a variable for the buffer where the service returns the path name of the directory.

return_value
(output,INT,4) is a variable where the service returns the length of the path name in the buffer if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Example

The following code gets the working directory for the caller. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
MVC   BUFLENA,=F'1024'      Max directory name return area
SPACE ,
CALL  BPX1GCW,              Get working directory name        +
      (BUFLENA,             Input: Length directory work area +
      BUFFERA,              Buffer                            +
      RETVAL,               Return value: 0 or -1             +
      RETCODE,              Return code                       +
      RSNCODE),             Reason code                       +
      VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The process did not have permission to read or search a component of the working directory's path name.
EINVAL	The *buffer_length* was specified as zero.
EIO	An input/output error occurred.
ENAMETOOLONG	The path name obtained by the routine is longer than 1023 characters.
ENOENT	A component of a path name does not exist. This will be returned if a component of the working directory path name was deleted.
ERANGE	The specified *buffer_length* is less than the length of the path name of the working directory.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Another callable service related to this service is:

- “chdir (BPX1CHD) — Change the Working Directory” on page 26

getegid (BPX1GEG) — Get the Effective Group ID

BPX1GEG

effective_group_ID

Purpose

Use the getegid (BPX1GEG) service to get the effective group ID (GID) of the calling process.

Parameters

effective_group_ID
(output,INT,4) is a variable where the service returns the effective group ID of the calling process.

Usage Note

If this service fails, the process abends.

Example

The following code gets the effective group ID of the caller. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         CALL  BPX1GEG,              Get the effective group ID        +
               (RETVAL),             Return value: effective group ID  +
               VL,MF=(E,PLIST)       ----------------------------------
```

Related Services

Other callable services related to this service are:

- “geteuid (BPX1GEU) — Get the Effective User ID” on page 115
- “getgid (BPX1GID) — Get the Real Group ID” on page 116
- “getuid (BPX1GUI) — Get the Real User ID” on page 141
- “setegid (BPX1SEG) — Set the Effective Group ID” on page 286
- “seteuid (BPX1SEU) — Set the Effective User ID” on page 288
- “setgid (BPX1SGI) — Set the Group ID” on page 290
- “setuid (BPX1SUI) — Set User IDs” on page 299.

geteuid (BPX1GEU) — Get the Effective User ID

BPX1GEU

effective_user_ID

Purpose

Use the geteuid (BPX1GEU) service to get the effective user ID (UID) of the calling process.

Parameters

effective_user_ID
(output,INT,4) is a variable where the service returns the effective user ID of the calling process.

Usage Note

If this service fails, the process abends.

Example

The following code gets the effective user ID of the caller. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551.

```
         CALL  BPX1GEU,             Get the effective user ID         +
               (RETVAL),            Return value: effective user ID   +
               VL,MF=(E,PLIST)      ----------------------------------
```

Related Services

Other callable services related to this service are:

- "getuid (BPX1GUI) — Get the Real User ID" on page 141
- "seteuid (BPX1SEU) — Set the Effective User ID" on page 288
- "setuid (BPX1SUI) — Set User IDs" on page 299.

getgid (BPX1GID) — Get the Real Group ID

BPX1GID

real_group_ID

Purpose

Use the getgid (BPX1GID) service to get the real group ID (GID) of the calling process.

Parameters

real_group_ID
(output,INT,4) is a variable where the service returns the real group ID.

Usage Note

If this service fails, the process abends.

Example

The following code gets the real group ID of the caller. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         CALL  BPX1GID,              Get the real group ID            +
               (RETVAL),             Return value: real group ID      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Related Services

Other callable services related to this service are:

- “getegid (BPX1GEG) — Get the Effective Group ID” on page 114
- “setegid (BPX1SEG) — Set the Effective Group ID” on page 286
- “setgid (BPX1SGI) — Set the Group ID” on page 290.

getgrgid (BPX1GGI) — Access the Group Database by ID

BPX1GGI

group_ID
return_value
return_code
reason_code

Purpose

Use the getgrgid (BPX1GGI) service to get information about a group and its members. You specify the group by the group ID (GID).

Parameters

group_ID
(input,INT,4) is a variable for specifying the ID of the group you want information about.

return_value
(output,INT,4) is a variable where the service returns an address, or 0.

If an entry for the specified group ID is found, *return_value* is set to the address of the BPXYGIDS macro. See "BPXYGIDS — Map the Data Structure Returned for the getgrnam and getgrgid Services" on page 425.

If no entry for the group ID is found, then *return_value* is set to 0.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is zero.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is zero.

Usage Notes

1. The return value points to data that may change or go away after the next getgrgid (BPX1GGI) or getgrnam (BPX1GGN) service request from this thread. Move data to the program's storage if it is needed for future reference.
2. If the same group ID is assigned to more than one group name, this function cannot distinguish which group is meant. Data is returned for one of the groups, but which group is unpredictable.
3. When called from REXX, only the first 500 members of a group are returned. A REXX exec must check the member count in the returned data structure to see if the data was truncated.
4. If multiple groups exist with the same group ID, one of the groups is selected, but which one is unpredictable.
5. To be authorized to obtain a group database entry, one of the following must be true:
   - The External Security Manager (ESM) grants the requestor authority to read the entry, or
   - An ESM is either not installed or defers authorization to CP, and:
     – The effective UID of the active process is 0, or
     – The real or effective GID of the active process matches the GID of the selected group, or
     – The requesting user is a member of the selected group, or

– The requesting VM user ID has the attribute POSIXOPT QUERYDB ALLOW set, either through a statement in its CP directory entry or through a specified or defaulted setting in the system configuration file that is not overridden in the directory entry.

Example

The following code accesses the group database by the ID of the caller and returns a structure identifying the groups by ID. The group ID value is set to 5. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYGIDS — Map the Data Structure Returned for the getgrnam and getgrgid Services” on page 425.

```
         MVC   GROUPID,=XL4'00000005'  Value of group ID
         SPACE ,
         CALL  BPX1GGI,              Access the group database         +
               (GROUPID,             Input: Group ID                   +
               RETVAL,               Return value: 0 or ->BPXYGIDS     +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
         ICM   R8,B'1111',RETVAL
         BZ    NOGIDS
         USING GIDS,R8
*        access the group structure
         DROP  R8
NOGIDS   EQU   *
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECMSERR	An internal error occurred during CMS processing. Consult the reason code to determine the exact reason the error occurred. For an out of storage condition, the reason code will be set to JrUnexpectedError. If the request to CP to obtain the group database information failed because no POSIX communication area was identified to CP, or the active PID in the POSIX communication area was not a PID allocated to this virtual configuration, or the buffer address provided to CP was invalid or protected against storing, the reason code will be JrInternalError.
ECPERR	An error occurred while retrieving information from CP. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JrCPNotFound, JrCPNotAvail, JrCPNotAuthorized and JrCPInternalError.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “getgrnam (BPX1GGN) — Access the Group Database by Name” on page 119
- “getlogin (BPX1GLG) — Get the User Login Name” on page 128.

getgrnam (BPX1GGN) — Access the Group Database by Name

BPX1GGN

group_name_length
group_name
return_value
return_code
reason_code

Purpose

Use the getgrnam (BPX1GGN) service to get information about a group and its members. You specify the group by name.

Parameters

group_name_length
(input,INT,4) is a variable for specifying the length of the *group_name* parameter.

group_name
(input,CHAR,*group_name_length*) is a variable for specifying the name of the group you want information about.

return_value
(output,INT,4) is a variable where the service returns an address, or 0.

If an entry for the specified group name is found in the group database, *return_value* is set to the address of the BPXYGIDS macro, which contains information about the group. See "BPXYGIDS — Map the Data Structure Returned for the getgrnam and getgrgid Services" on page 425.

If no entry for the group name is found, *return_value* is set to 0.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is zero.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is zero.

Usage Notes

1. The return value points to data that can change or go away after the next getgrnam (BPX1GGN) or getgrgid (BPX1GGI) call from this thread. Move data to your own dynamic storage if you need it for future reference.
2. When called from REXX, only the first 500 members of a group are returned. A REXX exec must check the member count in the returned data structure to see if the data was truncated.
3. To be authorized to obtain a group database entry, one of the following must be true:
   - The External Security Manager (ESM) grants the requestor authority to read the entry, or
   - An ESM is either not installed or defers authorization to CP, and
     - The effective UID of the active process is 0, or
     - The real or effective GID of the active process matches the GID of the designated group, or
     - The requesting user is a member of the designated group, or

– The requesting VM user ID has the attribute POSIXOPT QUERYDB ALLOW set, either through a statement in its CP directory entry or through a specified or defaulted setting in the system configuration file that is not overridden in the directory entry.

Example

The following code accesses the group database by the name of the caller and returns a structure identifying the groups by ID. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structure, see "BPXYGIDS — Map the Data Structure Returned for the getgrnam and getgrgid Services" on page 425.

```
         MVC   GRNAMELN,=F'7'
         MVC   GRPGMNAME(7),=CL7'EXTSERV'
         SPACE ,
         CALL  BPX1GGN,              Access the group database         +
               (GRNAMELN,            Input: Length of group name       +
               GRPGMNAME,            Input: Name of group              +
               RETVAL,               Return value: 0 or ->BPXYGIDS     +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECMSERR	An internal error occurred during CMS processing. Consult the reason code to determine the exact reason the error occurred. For an out of storage condition, the reason code will be set to JrUnexpectedError. If the request to CP to obtain the group database information failed because no POSIX communication area was identified to CP, or the active PID in the POSIX communication area was not a PID allocated to this virtual configuration, or the buffer address provided to CP was invalid or protected against storing, the reason code will be JrInternalError.
ECPERR	An error occurred while retrieving information from CP. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JrCPNotFound, JrCPNotAvail, JrCPNotAuthorized, and JrCPInternalError.
EINVAL	The *group_name_length* parameter is not valid.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, "Return Codes," on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, "Reason Codes," on page 495.

Related Services

Other callable services related to this service are:

- "getgrgid (BPX1GGI) — Access the Group Database by ID" on page 117
- "getlogin (BPX1GLG) — Get the User Login Name" on page 128.

getgroups (BPX1GGR) — Get a List of Supplementary Group IDs

BPX1GGR

group_ID_list_size
group_ID_list_pointer_address
return_value
return_code
reason_code

Purpose

Use the getgroups (BPX1GGR) service to get the number of supplementary group IDs (GIDs) for the calling process. Optionally, you can also get a list of those supplementary group IDs.

Parameters

group_ID_list_size
(input,INT,4) is a variable for specifying the number of fullword entries in the group ID list. This number must be at least as great as the total number of group IDs for the process, or 0.

Specifying 0 means that you want to receive only a count of the actual number of group IDs for the calling process and not the list of those IDs.

group_ID_list_pointer_address
(input,INT,4) is a variable for specifying the address of the storage area in which the service is to place the list of supplementary group IDs. If the request is successful, the storage area is an array of fullwords, each containing a supplementary group ID for the calling process.

If *group_ID_list_size* is specified as 0, *group_ID_list_pointer_address* is ignored and does not have to be set to a valid address.

return_value
(output,INT,4) is a variable where the service returns a count of supplementary group IDs, or -1:

- If *group_ID_list_size* is specified as 0, this is the total number of supplementary group IDs for the process.
- If *group_ID_list_size* is specified as greater than 0, this is the actual number of group IDs put into the area specified by *group_ID_list_pointer_address*.
- If an error is detected, a -1 is returned.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Example

The following code provides the caller with a list of supplementary group IDs. The code sets BUFW size to 256. The actual BUFW size is determined from the previous BPX1GGR RETVAL when BUFW was 0. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
*        MVC   BUFW,=XL4'00000256'   Value of buffer BUFW
         LA    R15,BUFFERA           Space for BUFW words
```

```
         ST    R15,BUFA              ->Array for group IDs
         SPACE ,
         CALL  BPX1GGR,              Get list of supplementary grp IDs +
               (BUFW,                Input: Group ID list size         +
               BUFA,                 ->Buffer for Group ID list address+
               RETVAL,               Return value: -1, 0, ID count     +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECMSERR	An internal error occurred during CMS processing. Consult the reason code to determine the exact reason the error occurred. If the request to CP to obtain the group membership information failed because no POSIX communication area was identified to CP, or the active PID in the POSIX communication area was not a PID allocated to this virtual configuration, the reason code will be JrInternalError.
ECPERR	An error occurred while retrieving information from CP. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JrCPNotAuthorized and JrCPInternalError.
EINVAL	The *group_ID_list_size* parameter was not equal to `0` and was less than the number of supplementary group IDs, or the *group_ID_list_pointer_address* was not valid. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JrBuffTooSmall and JrBadAddress.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Another callable service related to this service is:

- “setgid (BPX1SGI) — Set the Group ID” on page 290.

getgroupsbyname (BPX1GUG) — Get a List of Supplementary Group IDs by User Name

BPX1GUG

user_name_length
user_name
group_ID_list_size
group_ID_list_pointer_address
return_value
return_code
reason_code

Purpose

Use the getgroupsbyname (BPX1GUG) service to get the number of supplementary group IDs (GIDs) for a specified user name. Optionally, you can also get a list of those supplementary group IDs.

Parameters

user_name_length
(input,INT,4) is a variable for specifying the length of the *user_name* parameter.

user_name
(input,CHAR,*user_name_length*) is a variable for specifying the name of the user you want information about.

group_ID_list_size
(input,INT,4) is a variable for specifying the number of fullword entries in the group ID list. This number must be at least as great as the total number of group IDs for the process, or `0`.

Specifying `0` means that you want to receive only a count of the actual number of group IDs for the calling process and not the list of those IDs.

group_ID_list_pointer_address
(input,INT,4) is a variable for specifying the address of the storage area where the service is to place the list of supplementary group IDs as specified in the user database. If the request is successful, the storage is an array of fullwords, each containing a supplementary group ID for the specified user name.

If *group_ID_list_size* is specified as `0`, *group_ID_list_pointer_address* is ignored and does not have to be set to a valid address.

return_value
(output,INT,4) is a variable where the service returns the number of supplementary group IDs, or `-1`:

- If *group_ID_list_size* is specified as `0`, this is the total number of supplementary group IDs for the user.
- If *group_ID_list_size* is specified as greater than `0`, this is the actual number of group IDs put into the area specified by *group_ID_list_pointer_address*.
- If an error is detected, a `-1` is returned.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. The getgroupsbyname (BPX1GUG) service is not sensitive to the case of the *user_name* specified on input. This means that a *user_name* of BRIAN is considered the same as a *user_name* of brian.
2. To be authorized to obtain a supplementary group list for a user name, one of the following must be true:
   - The External Security Manager (ESM) grants the requestor authority to obtain the list, or
   - An ESM is either not installed or defers authorization to CP, and:
     - The UID of the specified user name matches the real or effective UID of the active process, or
     - The effective UID of the active process is 0, or
     - The requesting VM user ID has the attribute POSIXOPT QUERYDB ALLOW set, either through a statement in its CP directory entry or through a specified or defaulted setting in the system configuration file that is not overridden in the directory entry.

Example

The following code returns the number of supplementary group IDs, up to 9, for user Pebbles. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   USERNLEN,=F'7'
         MVC   USERNAME(07),=CL07'Pebbles'
         MVC   BUFLENA,=F'9'
         LA    R15,BUFFERA
         ST    R15,BUFA
         SPACE ,
         CALL  BPX1GUG,              Get list of groups by user name   +
               (USERNLEN,            Input: User name length           +
               USERNAME,             Input: User name                  +
               BUFLENA,              Input: Group ID list size         +
               BUFA,                 Group ID list address             +
               RETVAL,               Return value: -1, or # of grp IDs +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECMSERR	An internal error occurred during CMS processing. Consult the reason code to determine the exact reason the error occurred. If the request to CP to obtain the group database information failed because no POSIX communication area was identified to CP, or the active PID in the POSIX communication area was not a PID allocated to this virtual configuration, the reason code will be JrInternalError.
ECPERR	An error occurred while retrieving information from CP. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JrCPBadAddress, JrCPUserNotFound, JrCPNotAvail, JrCPNotAuthorized, and JrCPInternalError.

Return Code	Explanation
EINVAL	The *group_ID_list_size* parameter was not equal to 0 and was less than the number of supplementary group IDs; or the *user_name* or *user_name_length* fields were incorrect; or the *group_ID_list_pointer_address* was not valid. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JrOK, JrBuffTooSmall and JrBadAddress.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Another callable service related to this service is:

- “setgid (BPX1SGI) — Set the Group ID” on page 290.

gethostid/gethostname (BPX1HST) — Get ID or Name Information about a Socket Host

BPX1HST

domain
name_length
name
return_value
return_code
reason_code

Purpose

Use the gethostid/gethostname (BPX1HST) service to obtain the ID or name of the socket host.

Parameters

domain
(input,INT,4) is a variable for specifying a value that represents a communications domain. This must be AF_INET or AF_INET6. This value is defined in the BPXYSOCK macro. See "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465.

name_length
(input/output,INT,4) is a variable for specifying the length of the *name* parameter, which also indicates the type of request:

- If `0` is specified on input, the service returns the host ID in the *return_value* parameter. This is only supported for AF_INET sockets.
- If a nonzero value is specified on input, it represents the maximum length of the host name that is to be returned in the *name* parameter. The length should be less than 4096 bytes (4KB). On return, the service updates this field with the length of the name returned in *name*, including the trailing null.

name
(output,CHAR,*name_length*) is a variable where the service returns the host name, if a nonzero value was specified for *name_length*. This name is null-terminated if there is sufficient room in the buffer.

return_value
(output,INT,4) is a variable where the service returns one of the following:

- The host ID, if `0` was specified for *name_length*.
- `0`, if a nonzero *name_length* was specified and the name is successfully returned.
- `-1`, if the request is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Characteristics and Restrictions

These functions work only for AF_INET and AF_INET6 sockets, not AF_UNIX sockets or AF_IUCV sockets. These functions are not supported for IPv6 hosts.

Example

The following code requests the host ID and the host name for an AF_INET domain. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services” on page 465.

```
         XC    BUFLENA,BUFLENA
         CALL  BPX1HST,              Request host id                  +
               (=A(AF_INET),         Input: Domain - AF_INET          +
               BUFLENA,              Input: Length - No buffer - get id+
               BUFFERA,              Output: (not used with Length=0)  +
               RETVAL,               Return value: hostid or 0 or -1   +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

```
         MVC   BUFLENA,=A(L'BUFFERA)
         CALL  BPX1HST,              Request host name                +
               (=A(AF_INET),         Input: Domain - AF_INET          +
               BUFLENA,              Input: Length - for output name   +
               BUFFERA,              Output: Buffer for host name      +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EAFNOSUPPORT	The address family is not supported.
EAGAIN	The physical file system was unavailable. The following reason code can accompany this return code: JRPfsSuspend.
EIO	An I/O error occurred. The following reason code can accompany this return code: JRPfsDead.
ENOENT	The domain that was specified was found to be not active. The following reason code can accompany this return code: JRDomainNotSupported.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

getlogin (BPX1GLG) — Get the User Login Name

BPX1GLG

return_value

Purpose

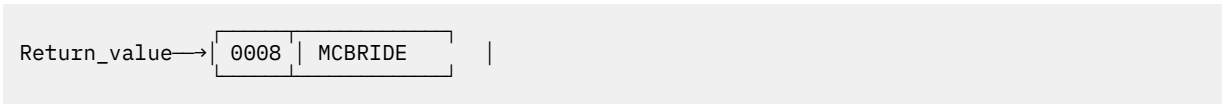
Use the getlogin (BPX1GLG) service to get the user login name associated with the current process.

Parameters

return_value

(output,INT,4) is a variable where the service returns a pointer to a login name field, or 0.

If a login name is found, *return_value* is set to the address of a field containing the login name length followed by the login name. The login name length is a fullword. For example:



If a login name is not found, *return_value* is set to 0.

Usage Note

This service returns a pointer to static data that will not be overwritten by a subsequent call. You should not store data into this area.

Example

The following code gets the login name of the caller. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         CALL  BPX1GLG,             Get the login name               +
               (RETVAL),            Returns value, 0 or ->login name  +
               VL,MF=(E,PLIST)      ---------------------------------
```

Related Services

Other callable services related to this service are:

- “geteuid (BPX1GEU) — Get the Effective User ID” on page 115
- “getpwnam (BPX1GPN) — Access the User Database by User Name” on page 132
- “getpwuid (BPX1GPU) — Access the User Database by User ID” on page 134
- “getuid (BPX1GUI) — Get the Real User ID” on page 141.

getpgrp (BPX1GPG) — Get the Process Group ID

BPX1GPG

group_ID

Purpose

Use the getpgrp (BPX1GPG) service to get the process group ID (PGID) of the calling process.

Parameters

group_ID
(output,INT,4) is a variable where the service places the caller's process group ID.

Example

The following code gets the process group ID of the caller. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551.

```
         CALL  BPX1GPG,              Get the process group ID          +
               (RETVAL),             Return value: group ID            +
               VL,MF=(E,PLIST)       ----------------------------------
```

Related Services

Other callable services related to this service are:

- "setpgid (BPX1SPG) — Set a Process Group ID for Job Control" on page 294
- "setsid (BPX1SSI) — Create a Session and Set the Process Group ID" on page 297.

getpid (BPX1GPI) — Get the Process ID

BPX1GPI

process_ID

Purpose

Use the getpid (BPX1GPI) service to get the process ID (PID) of the calling process.

Parameters

process_ID
(output,INT,4) is a variable where the service places the caller's process ID.

Example

The following code gets the process ID of the caller. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         CALL  BPX1GPI,              Get the process ID               +
               (RETVAL),             Returns value, Process ID        +
               VL,MF=(E,PLIST)       ---------------------------------
```

Related Services

Other callable services related to this service are:

- “exec (BPX1EXC) — Run a Program” on page 72
- “getppid (BPX1GPP) — Get the Parent Process ID” on page 131
- “kill (BPX1KIL) — Send a Signal to a Process” on page 146
- “spawn (BPX1SPN) — Spawn a Process” on page 333.

getppid (BPX1GPP) — Get the Parent Process ID

BPX1GPP

parent_process_ID

Purpose

Use the getppid (BPX1GPP) service to get the parent process ID (PPID) of the calling process.

Parameters

parent_process_ID
(output,INT,4) is a variable where the service returns the parent process ID of the calling process.

Example

The following code gets the process ID of the caller's parent. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         CALL  BPX1GPP,              Get PID of the parent process     +
               (RETVAL),             Returns value, parent's process ID+
               VL,MF=(E,PLIST)       ----------------------------------
```

Related Services

Other callable services related to this service are:

- “exec (BPX1EXC) — Run a Program” on page 72
- “getpid (BPX1GPI) — Get the Process ID” on page 130
- “kill (BPX1KIL) — Send a Signal to a Process” on page 146
- “spawn (BPX1SPN) — Spawn a Process” on page 333.

getpwnam (BPX1GPN) — Access the User Database by User Name

BPX1GPN

user_name_length
user_name
return_value
return_code
reason_code

Purpose

Use the getpwnam (BPX1GPN) service to get information about a user identified by name.

Parameters

user_name_length
(input,INT,4) is a variable for specifying the length of the *user_name* parameter.

user_name
(input,CHAR,*user_name_length*) is a variable for specifying the name of the user you want information about. This name is specified in the CP directory entry that defines the user to the system.

return_value
(output,INT,4) is a variable where the service returns an address, or 0:

- If an entry for the specified user name is found in the user database, *return_value* is set to the address of the BPXYGIDN macro, which contains information about the user. See "BPXYGIDN — Map the Data Structure Returned for the getpwnam and getpwuid Services" on page 424.
- If no entry for the user name is found, *return_value* is set to 0.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is zero.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is zero.

Usage Notes

1. The *return_value* points to data that may change or go away after the next getpwnam (BPX1GPN) or getpwuid (BPX1GPU) service request from this thread. Move data to your own dynamic storage if you need it for future reference.
2. The default initial user program is /bin/sh.
3. The default initial working directory is /.
4. The getpwnam (BPX1GPN) service is not sensitive to the case of the *user_name* specified on input. This means that a *user_name* of MEGAN is considered the same as a *user_name* of megan. However the user name returned in the database entry is in lower case.
5. To be authorized to obtain a user database entry, one of the following must be true:
   - The External Security Manager (ESM) grants the requestor authority to read the entry, or
   - An ESM is either not installed or defers authorization to CP, and:
     – The real or effective UID of the active process matches the UID of the designated user, or

- The effective UID of the active process is 0, or
- The requesting VM user ID has the attribute POSIXOPT QUERYDB ALLOW set, either through a statement in its CP directory entry or through a specified or defaulted setting in the system configuration file that is not overridden in the directory entry.

Example

The following code accesses the user database by the user ID of the caller and returns a structure identifying the user. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structure, see "BPXYGIDN — Map the Data Structure Returned for the getpwnam and getpwuid Services" on page 424.

```
         MVC   USERNLEN,=F'8'
         MVC   USERNAME(8),=CL8'Pebbles'
         SPACE ,
         CALL  BPX1GPN,              Access the user database          +
               (USERNLEN,            Input: Length of user name        +
               USERNAME,             Input: Name of user               +
               RETVAL,               Return value 0 or ->BPXYGIDN      +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECMSERR	An internal error occurred during CMS processing. Consult the reason code to determine the exact reason the error occurred. For an out of storage condition, the reason code will be set to JrUnexpectedError. If the request to CP to obtain the user database information failed because no POSIX communication area was identified to CP, or the active PID in the POSIX communication area was not a PID allocated to this virtual configuration, or the buffer address provided to CP was invalid or protected against storing, the reason code will be JrInternalError.
ECPERR	An error occurred while retrieving information from CP. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JrCPNotFound, JrCPNotAuthorized, JrCPNotAvail, and JrCPInternalError.
EINVAL	The *user_name_length* parameter is incorrect.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, "Return Codes," on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, "Reason Codes," on page 495.

Related Services

Other callable services related to this service are:

- "getlogin (BPX1GLG) — Get the User Login Name" on page 128
- "getpwuid (BPX1GPU) — Access the User Database by User ID" on page 134.

getpwuid (BPX1GPU) — Access the User Database by User ID

BPX1GPU

user_ID
return_value
return_code
reason_code

Purpose

Use the getpwuid (BPX1GPU) service to get information about a user identified by user ID (UID).

Parameters

user_ID
(input,INT,4) is a variable for specifying the user ID of the user you want information about.

return_value
(output,INT,4) is a variable where the returns an address, or 0:

- If an entry for the specified user ID is found in the user database, *return_value* is set to the address of the BPXYGIDN macro, which contains information about the user. See "BPXYGIDN — Map the Data Structure Returned for the getpwnam and getpwuid Services" on page 424.
- If no entry for the user ID is found, *return_value* is set to 0.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is zero.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is zero.

Usage Notes

1. The *return_value* points to data that can change or go away after the next getpwuid (BPX1GPU) or getpwnam (BPX1GPN) service request from this thread. Move data to the program's dynamic storage if the program needs it for future reference.
2. If the same user ID is defined for more than one VM user in the directory, this service cannot distinguish which one is meant. Data is returned about one of the users, but which one is unpredictable.
3. The default initial user program is /bin/sh.
4. The default initial working directory is /.
5. To be authorized to obtain a user database entry, one of the following must be true:
   - The External Security Manager (ESM) grants the requestor authority to read the entry, or
   - An ESM is either not installed or defers authorization to CP, and:
     – The real or effective UID of the active process matches the UID of the designated user, or
     – The effective UID of the active process is 0, or
     – The requesting VM user ID has the attribute POSIXOPT QUERYDB ALLOW set, either through a statement in its CP directory entry or through a specified or defaulted setting in the system configuration file that is not overridden in the directory entry.

Example

The following code accesses the user database by the user name of the caller and returns a structure identifying the user. The code sets the user ID value to 1. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structure, see "BPXYGIDN — Map the Data Structure Returned for the getpwnam and getpwuid Services" on page 424.

```
         MVC   USERID,=XL4'00000001  Value of user ID
         SPACE ,
         CALL  BPX1GPU,              Access database by user ID        +
               (USERID,              Input: User ID                    +
               RETVAL,               Return value 0 or ->BPXYGIDN      +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECMSERR	An internal error occurred during CMS processing. Consult the reason code to determine the exact reason the error occurred. For an out of storage condition, the reason code will be set to JrUnexpectedError. If the request to CP to obtain the user database information failed because no POSIX communication area was identified to CP, or the active PID in the POSIX communication area was not a PID allocated to this virtual configuration, or the buffer address provided to CP was invalid or protected against storing, the reason code will be JrInternalError.
ECPERR	An error occurred while retrieving information from CP. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JrCPNotFound, JrCPNotAuthorized, JrCPNotAvail, and JrCPInternalError.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, "Return Codes," on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, "Reason Codes," on page 495.

Related Services

Other callable services related to this service are:

- "getpwnam (BPX1GPN) — Access the User Database by User Name" on page 132
- "getuid (BPX1GUI) — Get the Real User ID" on page 141.

getsockname/getpeername (BPX1GNM) — Get the Name of a Socket or Peer

BPX1GNM

socket_descriptor
operation
sockaddr_length
sockaddr
return_value
return_code
reason_code

Purpose

Use the getsockname/getpeername (BPX1GNM) service to obtain the name of a socket or the name of a peer connected to a socket.

Parameters

socket_descriptor
(input,INT,4) is a variable for specifying the descriptor of the socket.

operation
(input,INT,4) is a variable for specifying a value that indicates the operation to be performed. These values are defined in the BPXYSOCK macro. See "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465.

sockaddr_length
(input/output,INT,4) is a variable for specifying the length of the *sockaddr* parameter. This value must be large enough to accommodate the maximum length of the SOCKADDR structure to be returned in *sockaddr*, but less than 4096 bytes (4KB). On output, the service updates this field with the size of the data returned in *sockaddr*.

sockaddr
(output,INT,*sockaddr_length*) is a variable where the service returns the SOCKADDR structure containing the socket name or peer name. This field is mapped by the BPXYSOCK macro. See "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Example

The following code gets the peer name, and then requests the socket name. SOCKDESC was returned by a previous call to socket (BPX1SOC). This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structure,

see “BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services” on page 465.

```
         SPACE ,
         CALL  BPX1GNM,                Get peername                    +
               (SOCKDESC,              Input: Socket Descriptor        +
               SOCK#GNMOPTGETPEERNAME, Input: Indicate getpeername     +
               SOCK#LEN+SOCK_SUN#LEN,  Input: Length - Sockaddr        +
               SOCKADDR,               Output: Sockaddr structure      +
               RETVAL,                 Return value: 0 or -1           +
               RETCODE,                Return code                     +
               RSNCODE),               Reason code                     +
               VL,MF=(E,PLIST)       ---------------------------------
         SPACE ,
         CALL  BPX1GNM,                Get sockname                    +
               (SOCKDESC,              Input: Socket Descriptor        +
               SOCK#GNMOPTGETSOCKNAME, Input: Indicate getsockname     +
               SOCK#LEN+SOCK_SUN#LEN,  Input: Length - Sockaddr        +
               SOCKADDR,               Output: Sockaddr structure      +
               RETVAL,                 Return value: 0 or -1           +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	The socket descriptor is incorrect. The following reason codes can accompany this return code: JRFileDesNotInUse, JRFileNotOpen.
EINVAL	The length specified by the *sockaddr_length* parameter is too small to allow the name to be returned. The following reason code can accompany this return code: JRSocketCallParmError.
ENOBUFS	Unable to obtain a buffer.
ENOTCONN	The getpeername operation was specified and the socket is not connected.
ENOTSOCK	*socket_descriptor* does not refer to a valid socket descriptor. The following reason code can accompany this return code: JRMustBeSocket.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

getsockopt/setsockopt (BPX1OPT) — Get or Set Socket Options

BPX1OPT

socket_descriptor
operation
level
option_name
option_data_length
option_data
return_value
return_code
reason_code

Purpose

Use the getsockopt/setsockopt (BPX1OPT) service to get or set the options that are associated with an AF_INET or AF_INET6 socket.

Parameters

socket_descriptor
(input,INT,4) is a variable for specifying the descriptor of the socket.

operation
(input,INT,4) is a variable for specifying a value that indicates the operation to be performed. These values are defined in the BPXYSOCK macro. See "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465.

level
(input,INT,4) is a variable for specifying the level for which the option is set or being set. These values are defined in the BPXYSOCK macro. See "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465.

option_name
(input,INT,4) is a variable for specifying a value that indicates the option name. These values are defined in the BPXYSOCK macro. See "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465.

option_data_length
(input/output,INT,4) is a variable for specifying the length of the *option_data* parameter. This value should be the maximum length that *option_data* could be on output, but less than 4096 bytes (4KB). On return from getsockopt, the service updates this field with the size of the data returned in *option_data*.

option_data
(input/output,CHAR,*option_data_length*) is a variable where, for getsockopt, the service returns the data associated with the socket. For setsockopt, this is a variable for specifying the data to be associated with the socket.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. The socket descriptor must refer to an open socket.
2. The level of support for this service depends on the particular socket stack you have installed. Some options might not be defined by the BPXYSOCK macro. Refer to the documentation for the product you are using to determine the socket options it supports.

Example

The following code gets and then sets socket options. SOCKDESC was returned on a previous call to the socket (BPX1SOC) service. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services” on page 465.

```
         MVC   BUFLENA,=A(L'BUFFERA)
         CALL  BPX1OPT,                  Get socket options            +
               (SOCKDESC,                Input: Socket Descriptor      +
               =A(SOCK#OPTOPTGETSOCKOPT), Input: Indicate Get socket   +
               SOCK#SOL_SOCKET,          Input: Level                  +
               SOCK#SO_TYPE,             Input: Option name            +
               BUFLENA,                  Input: Length - option value  +
               BUFFERA,                  Output: Option value          +
               RETVAL,                   Return value: 0 or -1         +
               RETCODE,                  Return code                   +
               RSNCODE),                 Reason code                   +
               VL,MF=(E,PLIST)       ----------------------------------
         SPACE ,
         MVC   BUFLENA,=A(4)             SO_OOBINLINE has length=4
         CALL  BPX1OPT,                  Set socket options            +
               (SOCKDESC,                Input: Socket Descriptor      +
               =A(SOCK#OPTOPTSETSOCKOPT), Input: Indicate set socket   +
               SOCK#SOL_SOCKET,          Input: Level                  +
               SOCK#SO_TYPE,             Input: Option name            +
               BUFLENA,                  Input: Length - option value  +
               SOCK#SO_OOBINLINE,        Input: Option value           +
               RETVAL,                   Return value: 0 or -1         +
               RETCODE,                  Return code                   +
               RSNCODE),                 Reason code                   +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EAFNOSUPPORT	The address family is not supported.
EBADF	The *socket_descriptor* is not a valid file descriptor.
EINVAL	An incorrect argument was supplied on the call. The following reason codes can accompany this return code: JRLevelNotSupp and JRBuffLenInvalid.
ENOBUFS	A buffer could not be obtained.
ENOPROTOOPT	The protocol or socket option is not available.
ENOSYS	The function is not implemented.
ENOTSOCK	*socket_descriptor* does not refer to a valid socket descriptor. The following reason code can accompany this return code: JRMustBeSocket.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

getuid (BPX1GUI) — Get the Real User ID

BPX1GUI

user_ID

Purpose

Use the getuid (BPX1GUI) service to get the real user ID (UID) of the calling process.

Parameters

user_ID
(output,INT,4) is a variable where the service returns the real user ID of the calling process.

Usage Note

If this service fails, the process abends.

Example

The following code gets the invoker's real user ID. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         CALL  BPX1GUI,              Get the real user ID             +
               (RETVAL),             Return value: real user ID       +
               VL,MF=(E,PLIST)       ---------------------------------
```

Related Services

Other callable services related to this service are:

- “geteuid (BPX1GEU) — Get the Effective User ID” on page 115
- “seteuid (BPX1SEU) — Set the Effective User ID” on page 288
- “setuid (BPX1SUI) — Set User IDs” on page 299.

givesocket (BPX1GIV) — Give a Socket to Another Program

```
BPX1GIV
        socket_descriptor
        client_ID
        return_value
        return_code
        reason_code
```

Purpose

Use the givesocket (BPX1GIV) service to make a specified socket available to a takesocket (BPX1TAK) call to be issued by another program.

Parameters

socket_descriptor
(input,INT,4) is a variable for specifying the descriptor of the socket to be given.

client_ID
(input/output,CHAR,length of BPXYCID macro) is a variable for specifying a data structure that contains client ID information identifying the program to which the socket is to be given. This information is typically obtained by the taking program with the getclientid (BPX1GCL) service and then passed to the server.

Client ID input may be as follows:

CIdDomain
Domain of the socket being given. These values are defined in the BPXYSOCK macro. See "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465.

CIdName
The VM user ID of the virtual machine that is running the target program, left-justified and padded with blanks. The target program can run in the same virtual machine as your program, in which case your program sets this field to its own VM user ID.

CIdTask
The "subtaskname" used by the target program when it established its IUCV connection with the TCP/IP server virtual machine. Alternatively, 8 blank characters may be specified to indicate that any application running in the target virtual machine can do the takesocket() call. If blanks are specified, the first application in the target virtual machine that issues the takesocket() call with the proper client ID of the giving program and proper socket number will become the new owner of the socket. Otherwise, only the application with the specified subtaskname will be authorized to take the socket.

CIdReserved
All binary zeros.

The client ID data structure is mapped by the BPXYCID macro. See "BPXYCID — Map the Client ID Structure" on page 415.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. Only the program identified by the client ID input of the givesocket service is allowed to take the socket using the takesocket (BPX1TAK) service.
2. The client ID output of getclientid (BPX1GCL) (issued by the secondary program and passed to the server) is intended to be used as the input client ID of the givesocket service.
3. If the given socket is not closed, it can still be used, even after the takesocket (BPX1TAK) has been done. The socket can be shared between the giver and taker in the same way that an inherited socket can be shared between parent and child after an exec (BPX1EXC) has been issued.
4. If the caller of givesocket issues the close (BPX1CLO) some time later, it may be necessary to coordinate with the caller of takesocket (BPX1TAK). The close itself does not interfere with takesocket, but if additional sockets are accepted, given away, and closed before takesocket is called, there can be several given sockets with the same descriptor that are waiting to be taken. This can cause unpredictable results.

   To avoid this problem, you can issue the select (BPX1SEL) service for a given socket, and the program can find out from select when the takesocket (BPX1TAK) call has been issued and it is safe to call close (BPX1CLO). For a general server, though, this is a very poorly performing design. Selecting on the main socket and having all given sockets wait for another connection or for one of the given sockets to be taken is very expensive, and should be avoided.

Example

The following code gives a socket to the program identified by CID (client ID). The target program may then use the takesocket (BPX1TAK) service to take the socket. SOCKDESC was previously set by a call to the accept (BPX1ACP) service. CID is set by the getclientid (BPX1GCL) service. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYCID — Map the Client ID Structure” on page 415.

```
         CALL  BPX1GIV,              give a socket to another program  +
               (SOCKDESC,            Input: Socket descriptor          +
               CID,                  Input: Clientid of recipient      +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EAFNOSUPPORT	The address family is not supported.
EBADF	The socket descriptor is not valid, or the socket has already been given. The following reason code can accompany this return code: JRFileDesNotInUse.
EBUSY	Listen() has been called for the socket (that is, it is not an active socket).
ECMSERR	A CMS environmental or internal error has occurred. The following reason code can accompany this return code: JRLockErr.

Return Code	Explanation
EINVAL	The *client_ID* parameter does not specify a valid client identifier, or the CIdDomain in the *client_ID* parameter does not match the actual domain of the input socket descriptor. The following reason code can accompany this return code: JRSocketCallParmError.
ENOTCONN	The socket is not in the connected state.
ENOTSOCK	The descriptor is for a file, not for a socket.
EOPNOTSUPP	The operation is not supported for the socket protocol.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “getclientid (BPX1GCL) — Obtain the Calling Program's Identifier” on page 110
- “takesocket (BPX1TAK) — Acquire a Socket from Another Program” on page 350

isatty (BPX1ITY) — Determine If a File Descriptor Represents a Terminal

BPX1ITY

file_descriptor

return_value

Purpose

Use the isatty (BPX1ITY) service to determine if a file is a terminal. You identify the file by file descriptor.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the file descriptor of the file.

return_value
(output,INT,4) is a variable where the service returns 1 if *file_descriptor* is a terminal, or 0 if it is not a terminal.

Usage Note

This function does not return a return value of -1. If the file descriptor is not valid, a zero is returned. If the service fails for other reasons, the process abends.

Example

The following code determines if the standard output device is a terminal. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551.

```
         CALL  BPX1ITY,              Determine if device is a TTY      +
               (=A(STDOUT_FILENO),   Input: File descriptor            +
               RETVAL),              Return value: 0 isn't, 1 is       +
               VL,MF=(E,PLIST)       ---------------------------------
         ICM   R15,B'1111',RETVAL    Test RETVAL
         BZ    PSEUDO                RETVAL=0 means device not terminal
```

Related Services

Another callable service related to this service is:

- "ttyname (BPX1TYN) — Get the Name of a Terminal" on page 373.

kill (BPX1KIL) — Send a Signal to a Process

BPX1KIL

process_ID
signal
signal_options
return_value
return_code
reason_code

Purpose

Use the kill (BPX1KIL) service to send a signal to a process or a process group.

Parameters

process_ID
(input,INT,4) is a variable for specifying the process ID (PID) of the process or process group you want to send a signal to:

- If *process_ID* is greater than 0, it is assumed to be a process ID. The signal is sent to the process with that specific process ID.
- If *process_ID* is 0, the signal is sent to all processes having a process group ID equal to that of the caller, and for which the caller has permission to send a signal.
- If *process_ID* is -1, the service returns a return value of -1 and return code ESRCH.
- If *process_ID* is less than -1, its absolute value is assumed to be a process group ID. The signal is sent to all processes having a process group ID equal to that absolute value, and for which the sender has permission to send a signal.

For more information, see "Characteristics and Restrictions" on page 147.

signal
(input,INT,4) is a variable for specifying the signal number to be sent to the processes indicated by the *process_ID* parameter. The signal number must be defined in the BPXYSIGH macro, or 0. The possible signals are shown in "Signal Defaults" on page 561.

If the signal is 0, error checking takes place but no signal is sent. Use a signal value of 0 to verify that the *process_ID* parameter is correct before actually sending a signal. However, this method does not verify permission to send the signal to the specified process ID.

signal_options
(input,INT,4) is a variable for specifying the binary flags that describe how the signal is to be handled by OpenExtensions and the user-supplied signal interface routine (SIR). This byte of user information is passed to the SIR in a data structure mapped by the BPXYPPSD macro. See "BPXYPPSD — Map the Signal Delivery Data Structure" on page 451. The *signal_options* parameter is mapped as follows:

First 2 bytes
User-defined bytes delivered with the signal to the SIR in the signal information control block. These bytes are mapped by PPSDKILDATA.

Last 2 bytes
Reserved

return_value
(output,INT,4) is a variable where the service returns 0 if it has permission to send the specified signal to any of the processes specified by the *process_ID* parameter. A return value of 0 means that the

signal was sent (or could have been sent, if the signal value was 0) to at least one of the specified processes.

If the signal could not be sent, -1 is returned.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Characteristics and Restrictions

1. The range of a signal is limited to processes in the same virtual machine as the caller; signals may not be sent to processes in other virtual machines.
2. A caller can send a signal if the real or saved set user ID of the caller is the same as the real or saved set user ID of the intended recipient. A caller can also send signals if the caller is a superuser.
3. Regardless of user ID, a caller can always send a SIGCONT signal to a process that is a member of the same session as the sender.
4. A caller can also send a signal to itself. If the signal is not blocked, at least one pending unblocked signal is delivered to the sender before the service returns control. Provided that no other unblocked signals are pending, the signal delivered is the signal sent. See Appendix E, “The Relationship of OpenExtensions Signals to Callable Services,” on page 557 for more information.

Example

The following code sends a signal (SIGUSR1) to all processes (for which access is allowed) in the invoker's process group. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYSIGH — Map Signal Constants” on page 462.

```
         MVC   PROCID,=A(0)          Invoker's process group
         CALL  BPX1KIL,              Send a signal to a process        +
               (PROCID,              Input: Process ID                 +
               =A(SIGUSR1#),         Input: Signal            BPXYSIGH +
               =A(0),                Input: Signal options             +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EINVAL	The value of *signal* is incorrect or is not the number of a supported signal.
EPERM	The caller does not have permission to send the signal to any process specified by the *process_ID* parameter.
ESRCH	No processes or process groups corresponding to *process_ID* were found.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “getpid (BPX1GPI) — Get the Process ID” on page 130
- “setpgid (BPX1SPG) — Set a Process Group ID for Job Control” on page 294
- “setsid (BPX1SSI) — Create a Session and Set the Process Group ID” on page 297
- “sigaction (BPX1SIA) — Examine or Change a Signal Action” on page 315.

link (BPX1LNK) — Create a Link to a File

```
BPX1LNK
        filename_length
        filename
        link_name_length
        link_name
        return_value
        return_code
        reason_code
```

Purpose

Use the link (BPX1LNK) service to create a link to a file. A link is a new name identifying an existing file. The new name does not replace the old one, but provides an additional way to refer to the file. To rename an existing file, see "rename (BPX1REN) — Rename a File or Directory" on page 251.

Parameters

filename_length
(input,INT,4) is a variable for specifying the length of the *filename* parameter.

filename
(input,CHAR,*filename_length*) is a variable for specifying the name of the existing file to which a link is to be established.

link_name_length
(input,INT,4) is a variable for specifying the length of the *link_name* parameter.

***link_name*,**
(input,CHAR,*link_name_length*) is a variable for specifying the name of the link.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. The link (BPX1LNK) service creates a link named *link_name* to an existing file named *filename*. This provides an alternate path name for the existing file, so the file can be accessed by the old name or the new name. The link can be stored in the same directory as the original file, or in a different directory.
2. If the link is created successfully, the service increments the link count of the file. The link count shows how many links exist for a file. (If the link is not created successfully, the link count is not incremented.)
3. Links are allowed only to files, not to directories.
4. If the link is created successfully, the change time of the linked-to file is updated. The change and modification times of the directory that holds the link are also updated.

Example

The following code creates a new way for **usr/dataproc.next.t** to link to an existing file, **usr/user05/yearrecs.t**. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551.

```
         MVC   BUFLENA,=F'21'
         MVC   BUFFERA(21),=CL21'usr/user05/yearrecs.t'
         MVC   BUFLENB,=F'19'
         MVC   BUFFERB(19),=CL19'usr/dataproc.next.t'
         SPACE ,
         CALL  BPX1LNK,              Create a link to a file           +
               (BUFLENA,             Input: Name length: existing      +
               BUFFERA,              Input: Name of existing file      +
               BUFLENB,              Input: Name length: link          +
               BUFFERB,              Input: Name of link to file       +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The process did not have appropriate permissions to create the link. Possible reasons include: • No search permission for a path name component of *filename* or *link_name* • No write permission for the directory intended to contain the link • No permission to access *filename*.
EEXIST	A file, directory, or symbolic link named *link_name* already exists. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRLnkNewPathExists.
EINVAL	The *filename* or *link_name* parameter is incorrect because it contains a null.
ELOOP	A loop exists in symbolic links encountered during resolution of the *filename* or *link_name* argument. This error is issued if more than 8 symbolic links are detected in the resolution of the file name or link name.
EMLINK	The file specified by *filename* already has its maximum number of links. The maximum number is LINK_MAX. The value of LINK_MAX can be determined through the pathconf (BPX1PCF) or fpathconf (BPX1FPC) service. See "pathconf (BPX1PCF) — Determine Configurable Path Name Variables Using Path Name" on page 194 or "fpathconf (BPX1FPC) — Determine Configurable Path Name Variables Using a Descriptor" on page 99.
ENAMETOOLONG	The *filename* or *link_name* parameter is longer than 1023 characters, or some component of the path name is longer than 255 characters. CMS does not support name truncation.
ENOENT	A component of the path name specified by *filename* or *link_name* was not found; or the file specified by *filename* was not found; or one of the two arguments is missing. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRLnkNoEnt.
ENOSPC	The directory intended to contain the link cannot be extended to contain another entry.

Return Code	Explanation
ENOTDIR	A path name component of one of the arguments is not a directory.
EPERM	The *filename* parameter contains the name of a directory, and links to directories are not allowed. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRLnkDir.
EROFS	Creating the link would require writing on a read-only file system. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRLnkROFileset.
EXDEV	The *filename* and *link_name* are on different file systems. CMS does not support links between file systems. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRLnkAcrossFilesets.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “rename (BPX1REN) — Rename a File or Directory” on page 251
- “unlink (BPX1UNL) — Remove a Directory Entry” on page 379.

listen (BPX1LSN) — Prepare a Server Socket to Queue Incoming Connection Requests from Clients

BPX1LSN

socket_descriptor
backlog
return_value
return_code
reason_code

Purpose

Use the listen (BPX1LSN) service to create a connection request queue for a server socket to queue incoming connection requests from a client.

Listen is used for connection-oriented sockets only. If a connection request arrives with the backlog queue full, the client may receive an ECONNREFUSED return code.

Parameters

socket_descriptor
(input,INT,4) is a variable for specifying the descriptor of the server socket.

backlog
(input,INT,4) is a variable for specifying the maximum length of the connection queue. For AF_INET and AF_INET6 sockets, if the backlog is greater than SOMAXCONN, this field is set to SOMAXCONN. If *backlog* is less than 0, *backlog* is set to 0.

For AF_UNIX and AF_IUCV sockets, this parameter is ignored.

return_value
(output,INT,4) is a variable where the service returns 0 if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Note

If a bind (BPX1BND) is not called before the listen request, listen returns an EINVAL return code.

Example

The following code issues a listen on a socket that was previously created with socket (BPX1SOC) and given a unique local name with bind (BPX1BND). SOCKDESC was returned from the call to BPX1SOC. Set the backlog count to 5. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services” on page 465.

```
         CALL  BPX1LSN,             Listen on a socket              +
               (SOCKDESC,           Input: Socket Descriptor        +
               =A(5),               Input: Backlog count of 5       +
```

```
                RETVAL,              Return value: 0 or -1            +
                RETCODE,             Return code                      +
                RSNCODE),            Reason code                      +
                VL,MF=(E,PLIST)      ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	The socket descriptor is incorrect. The following reason codes can accompany this return code: JRFileDesNotInUse, JRFileNotOpen.
EINVAL	An incorrect argument was supplied. The socket is not named (a bind has not been done), or the socket is ready to accept connections (a listen has already been done).
ENOBUFS	A buffer could not be obtained.
ENOTSOCK	*socket_descriptor* does not refer to a valid socket descriptor. The following reason code can accompany this return code: JRMustBeSocket.
EOPNOTSUPP	The socket descriptor specified a *datagram* socket. The listen service is valid only for *stream* sockets.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

lseek (BPX1LSK) — Change the File Offset

BPX1LSK

file_descriptor
offset
reference_point
return_value
return_code
reason_code

Purpose

Use the lseek (BPX1LSK) service to change the file offset to a new position. The file offset is the position in a file from which data is next read, or to which data is next written. The file is identified by its file descriptor.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the file descriptor for the file whose offset you want to change. The file descriptor is returned from the open (BPX1OPN) service.

offset
(input/output,INT,8) is a variable for specifying a signed number to indicate the offset change. The numeric part of the value is the amount (number of bytes) by which you want to change the offset; the sign indicates whether you want the offset to be moved forward in the file or backward.

This parameter is a doubleword to accommodate large files. For processing a singleword value, propagate the sign bit through the second word, so the final doubleword value has a valid sign.

On successful completion, the service returns the new file offset.

reference_point
(input,INT,4) is a variable for specifying the point where the offset is calculated from. The possible values are mapped by the BPXYSEEK macro. See "BPXYSEEK — Map Constants for the lseek Service" on page 455.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

If the request is successful, the new file offset is returned in the *offset* parameter.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. The *offset* parameter gives the length and direction of the offset change. The *reference_point* parameter states where the change is to start. For example, assume that a file is 2000 bytes long, and that the current file offset is 1000:

Offset Specified	Reference Point	New File Offset
80	SEEK_CUR	1080
1200	SEEK_SET	1200
-80	SEEK_END	1920
132	SEEK_END	2132

2. The file offset can be moved beyond the end of the file. If data is written at the new file offset, there will be a gap between the old end of the file and the start of the new data. A request to read data from anywhere within that gap completes successfully, and returns bytes with the value of zero in the buffer and the actual number of bytes read.

   Seeking alone, however, does not extend the file. Only if data is written at the new offset does the length of the file change.

Example

The following code changes the file (FILEDESC) offset to 80 bytes past the current offset. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYSEEK — Map Constants for the lseek Service” on page 455.

```
         MVC   FILEDESC,..           File descriptor from open
         MVC   OFFSET(08),=D'80'     Forward 80 Bytes
         MVC   REFPT,=A(SEEK_CUR)    Current offset of the file
         SPACE ,
         CALL  BPX1LSK,              Change a file's offset           +
               (FILEDESC,            File descriptor                  +
               OFFSET,               I/O: Offset in file              +
               REFPT,                Input: Reference point, BPXYSEEK +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	The *file_descriptor* parameter does not specify a valid, open file.
EINVAL	The *reference_point* parameter contained something other than one of the three options, or the combination of the *offset* and *reference_point* parameters would have placed the file offset before the beginning of the file. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRLskOffsetInvalid and JRLskWhenceIsInvalid.
ESPIPE	The *file_descriptor* refers to a pipe or a FIFO special file. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRLskOnPipe.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “fcntl (BPX1FCT) — Control Open File Descriptors” on page 88
- “open (BPX1OPN) — Open a File” on page 181
- “read (BPX1RED) — Read from a File or Socket” on page 228
- “sigaction (BPX1SIA) — Examine or Change a Signal Action” on page 315
- “write (BPX1WRT) — Write to a File or Socket” on page 401.

lstat (BPX1LST) — Get Status Information about a File or Symbolic Link by Path Name

BPX1LST

pathname_length
pathname
status_area_length
status_area
return_value
return_code
reason_code

Purpose

Use the lstat (BPX1LST) service to obtain status information about a file identified by its path name. This service is identical to the stat (BPX1STA) service, except when the specified path name is a symbolic link, which is a pointer to another file or directory. In this case, the status information returned by lstat (BPX1LST) relates to the symbolic link, rather than the file the symbolic link refers to. The stat service is explained in "stat (BPX1STA) -- Get Status Information about a File by Path Name" on page 340.

For the corresponding service using a file descriptor, see "fstat (BPX1FST) -- Get Status Information about a File by Descriptor" on page 102.

Parameters

pathname_length
(input,INT,4) is a variable for specifying the length of the *pathname* parameter.

pathname
(input,CHAR,*pathname_length*) is a variable for specifying the path name of the file for which you want to obtain status. See "Understanding Byte File System (BFS) Path Name Syntax" on page 6.

status_area_length
(input,INT,4) is a variable for specifying the length of the *status_area* parameter. To determine the value of *status_area_length*, use the BPXYSTAT macro. See "BPXYSTAT — Map the File Status Structure for the stat Service" on page 473. If the specified length is too small, the data returned in *status_area* is truncated.

status_area
(input,CHAR,length of BPXYSTAT macro or *status_area_length*, whichever is less) is a variable for the area where the service returns the status information for the file. This area is mapped by the BPXYSTAT macro. See "BPXYSTAT — Map the File Status Structure for the stat Service" on page 473.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(input,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. All modified data in the file identified by *pathname* is written to permanent storage when this service is requested. See "fsync (BPX1FSY) — Write Changes to Direct-Access Storage" on page 106.
2. All time fields in *status_area* are in POSIX format.
3. The File Mode field in *status_area* is mapped by the BPXYMODE macro. See "BPXYMODE — Map Mode Constants" on page 437. For information on the values for file type, see "BPXYFTYP — Map File Type Definitions" on page 423.

Characteristics and Restrictions

To obtain information about a file, you need not have permissions for the file itself; however, you must have search permission for all of the directory components of the path name.

Example

The following code obtains the file status for the file described by the symbolic name **labrec/sym**. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structure, see "BPXYSTAT — Map the File Status Structure for the stat Service" on page 473.

```
*   symbolic name established using symlink (BPX1SYM) system call
         MVC   BUFFERA(10),=CL10'labrec/sym'
         MVC   BUFLENA,=F'10'
         SPACE ,
         CALL  BPX1LST,              Get file status                  +
               (BUFLENA,             Input: Pathname length           +
               BUFFERA,              Input: Pathname                  +
               STATL,                Input: Length of buffer needed   +
               STAT,                 Buffer, mapped by BPXYSTAT       +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The process does not have permission to search some component of the *pathname* parameter.
ECMSERR	An internal error occurred.
EINVAL	Parameter error—for example, a zero-length buffer. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRBuffTooSmall.
ELOOP	A loop exists in symbolic links encountered during resolution of the *pathname* argument. This error is issued if more than 8 symbolic links are detected in the resolution of the path name.
ENAMETOOLONG	The *pathname* parameter is longer than 1023 characters, or some component of the path name is longer than 255 characters. This could be as a result of encountering a symbolic link during resolution of *pathname*, and the substituted string is longer than 1023 characters.
ENODEV	An attempt was made to use a character special file for a device not supported by OpenExtensions.

Return Code	Explanation
ENOENT	No file named *pathname* was found, or a path name was not specified. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRFileNotThere.
ENOTDIR	A component of the path name is not a directory.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “chmod (BPX1CHM) — Change the Mode of a File or Directory by Path Name” on page 28
- “exec (BPX1EXC) — Run a Program” on page 72
- “fpathconf (BPX1FPC) — Determine Configurable Path Name Variables Using a Descriptor” on page 99
- “fstat (BPX1FST) -- Get Status Information about a File by Descriptor” on page 102
- “link (BPX1LNK) — Create a Link to a File” on page 149
- “mkdir (BPX1MKD) — Make a Directory” on page 160
- “open (BPX1OPN) — Open a File” on page 181
- “pipe (BPX1PIP) — Create an Unnamed Pipe” on page 199
- “read (BPX1RED) — Read from a File or Socket” on page 228
- “symlink (BPX1SYM) — Create a Symbolic Link to a Path Name” on page 345
- “unlink (BPX1UNL) — Remove a Directory Entry” on page 379
- “utime (BPX1UTI) -- Set File Access and Modification Times” on page 382
- “write (BPX1WRT) — Write to a File or Socket” on page 401.

mkdir (BPX1MKD) — Make a Directory

BPX1MKD

pathname_length
pathname
mode
return_value
return_code
reason_code

Purpose

Use the mkdir (BPX1MKD) service to create a new, empty directory.

Parameters

pathname_length
(input,CHAR,4) is a variable for specifying the length of the *pathname* parameter.

pathname
(input,CHAR,*pathname_length*) is a variable for specifying the full path name of the directory to be created. See "Understanding Byte File System (BFS) Path Name Syntax" on page 6.

mode
(input,INT,4) is a variable for specifying the mode of the directory, which includes the file type and the permissions you grant to yourself, to your group, and to any user.

The file type is identified using the BPXYFTYP macro. Permissions are specified with the BPXYMODE macro. See "BPXYFTYP — Map File Type Definitions" on page 423 and "BPXYMODE — Map Mode Constants" on page 437.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. The file permission bits specified through the *mode* parameter are modified by the file creation mask of the calling process. (See "umask (BPX1UMK) — Set or Return the File Mode Creation Mask" on page 374). They are then used to set the file permission bits of the new directory.
2. The new directory's owner ID is set to the effective user ID (UID) of the calling process.
3. The mkdir (BPX1MKD) service sets the access, change, and modification times for the new directory. It also sets the change and modification times for the directory that contains the new directory.

Example

The following code creates a new and empty directory path name of **/usr/newprots/** with user read-execute, group write, other read-execute permissions. This example follows the rules of reentrancy. For

linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYFTYP — Map File Type Definitions” on page 423 and “BPXYMODE — Map Mode Constants” on page 437.

```
         MVC   BUFFERA(14),=CL14'/usr/newprots/'
         MVC   BUFLENA,=F'14'
         XC    S_MODE,S_MODE
         MVI   S_MODE2,S_IRUSR       Read search write read search
         MVI   S_MODE3,S_IXUSR+S_IWGRP+S_IROTH+S_IXOTH
         SPACE ,
         CALL  BPX1MKD,              Make a directory                 +
               (BUFLENA,             Input: Pathname length           +
               BUFFERA,              Input: Pathname                  +
               S_MODE,               Input: BPXYMODE and BPXYFTYP     +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The process did not have search permission on some component of *pathname*, or did not have write permission on the parent directory of the directory to be created.
EEXIST	There is already a file or directory with the specified path name. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRMkDirExist.
ELOOP	A loop exists in symbolic links encountered during resolution of the *pathname* argument. This error is issued if more than 8 symbolic links are detected in the resolution of the path name.
EMLINK	The link count of the parent directory has already reached the maximum defined for the system. Refer to the LINK_MAX in “pathconf (BPX1PCF) — Determine Configurable Path Name Variables Using Path Name” on page 194, or to “fpathconf (BPX1FPC) — Determine Configurable Path Name Variables Using a Descriptor” on page 99.
ENAMETOOLONG	The *pathname* parameter contains more than 1023 characters, or a component of the path name is longer than 255 characters.
ENOENT	Some component of *pathname* does not exist, or the *pathname* parameter is blank.
ENOSPC	The file system does not have enough space to contain a new directory, or the parent directory cannot be extended.
ENOTDIR	A component of the path name is not a directory.
EROFS	The parent directory of the directory to be created is on a read-only file system. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRMkDirROnly.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “chmod (BPX1CHM) — Change the Mode of a File or Directory by Path Name” on page 28
- “stat (BPX1STA) -- Get Status Information about a File by Path Name” on page 340
- “umask (BPX1UMK) — Set or Return the File Mode Creation Mask” on page 374.

mknod (BPX1MKN) — Make a FIFO or Character Special File

BPX1MKN

pathname_length
pathname
mode
device_identifier
return_value
return_code
reason_code

Purpose

Use the mknod (BPX1MKN) service to create a new character special file or FIFO special file (named pipe).

Parameters

pathname_length
(input,INT,4) is a variable for specifying the length of the *pathname* parameter.

pathname
(input,CHAR,*pathname_length*) is a variable for specifying the pathname of the special file to be created. See "Understanding Byte File System (BFS) Path Name Syntax" on page 6.

mode
(input,INT,4) is a variable for specifying the mode of the file, which includes the file type and the permissions you grant to yourself, to your group, and to any user. Specify the file type with the BPXYFTYP mapping macro and specify permissions with the BPXYMODE mapping macro. See "BPXYFTYP — Map File Type Definitions" on page 423 and "BPXYMODE — Map Mode Constants" on page 437.

device_identifier
(input,INT,4) is a variable for specifying a device identifier, or `0`. Specify *device_identifier* if you are creating a character special file. If a FIFO file is being created (mode file type specified as 4), then *device_identifier* is ignored.

The high-order 16 bits of *device_identifier* is the device major number. The device major number corresponds to a device driver supporting a class of devices—for example, interactive terminals. The low-order 16 bits of *device_identifier* is the device minor number. The device minor number corresponds to a specific device within the class of devices referred to by the device major number.

The device major numbers currently defined for use by OpenExtensions services are:

3. **/dev/tty**
4. **/dev/null**

For device major numbers 3 and 4, the device minor number is ignored.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. The file permission bits of *mode* are modified by the process's file creation mask and then used to set the file permission bits of the file being created. (See "umask (BPX1UMK) — Set or Return the File Mode Creation Mask" on page 374.)
2. The file's owner ID is set to the process's effective user ID (UID). The group ID is set to the group ID (GID) of the directory containing the file.
3. The mknod (BPX1MKN) service sets the access, change, and modification times for the new file. It also sets the change and modification times for the directory that contains the new file.

Characteristics and Restrictions

When the mknod (BPX1MKN) service is invoked to create a character special file, it is a privileged operation and requires superuser authority.

Example

The following code creates a FIFO (pipe) named **/u/fifos/fifio1** and user read-write, group read, other read permissions. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structure, see "BPXYFTYP — Map File Type Definitions" on page 423 and "BPXYMODE — Map Mode Constants" on page 437.

```
         MVC   BUFFERA(14),=CL14'/u/fifos/fifo1'
         MVC   BUFLENA,=F'14'
         XC    S_MODE,S_MODE
         MVI   S_TYPE,FT_FIFO        First in - first out
         MVI   S_MODE2,S_IRUSR       Read write read read
         MVI   S_MODE3,S_IWUSR+S_IRGRP+S_IROTH
         SPACE ,
         CALL  BPX1MKN,              Create FIFO or character special f+
               (BUFLENA,             Input: Pathname length            +
               BUFFERA,              Input: Pathname                   +
               S_MODE,               Input: BPXYMODE and BPXYFTYP      +
               =A(0),                Input: Device id not used here    +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The process does not have permission to search some component of *pathname*, or does not have write permission for the directory of the file to be created.
EEXIST	A file or directory named *pathname* already exists. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRSpFileExists.
EINVAL	The file type specified in the *mode* parameter is not 2 or 4. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRMknodInvalidType.

Return Code	Explanation
ELOOP	A loop exists in symbolic links encountered during resolution of the *pathname* argument. This error is issued if more than 8 symbolic links are detected in the resolution of the pathname.
ENAMETOOLONG	The *pathname* parameter is longer than 1023 characters, or a component of the pathname is longer than 255 characters.
ENOENT	A component of *pathname* was not found, or no pathname was specified. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JREndingSlashMknod.
ENOTDIR	A component of *pathname* is not a directory.
EROFS	The directory of the file is on a read-only file system. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRReadOnlyFilesetMknodReq.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “chmod (BPX1CHM) — Change the Mode of a File or Directory by Path Name” on page 28
- “exec (BPX1EXC) — Run a Program” on page 72
- “pipe (BPX1PIP) — Create an Unnamed Pipe” on page 199
- “stat (BPX1STA) -- Get Status Information about a File by Path Name” on page 340
- “umask (BPX1UMK) — Set or Return the File Mode Creation Mask” on page 374.

mount (BPX1MNT) — Make a File System Available

BPX1MNT

mountpoint_length
mountpoint_name
file_system_name_length
file_system_name
file_system_type
mount_mode
parm_length
parm
return_value
return_code
reason_code

Purpose

Use the mount (BPX1MNT) service to mount a local or remote file system, making the files in it available for use.

Parameters

mountpoint_length
(input,INT,4) is a variable for specifying the length of the *mountpoint_name* parameter.

mountpoint_name
(input,CHAR,*mountpoint_length*) is a variable for specifying the name of the mount point.

file_system_name_length
(input,INT,4) is a variable for specifying the length of the *file_system_name* parameter.

file_system_name
(input,CHAR,file_system_name_length) is a variable for specifying the name of the file system that is to be mounted. The file system name can be a Byte File System (BFS) path name or a Network File System (NFS) path name. See usage note "3" on page 167.

file_system_type
(input,CHAR,8) is a variable for specifying the file system type. For a byte file system, use VMBFS. For a network file system, use BPXFSNFS.

mount_mode
(input,INT,4) is a variable for specifying binary flags that show the mount mode (read or read/write).

This parameter is mapped by the BPXYMTM macro. See "BPXYMTM — Map the Modes for the mount and umount Services" on page 445.

parm_length
(input,INT,4) is a variable for specifying the length of the *parm* parameter.

parm
(input,CHAR,*parm_length*) is a variable for specifying file-system-specific parameters. These have a maximum length of 1024 bytes. See usage note "3" on page 167.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. The mount service effectively creates a virtual file system. After a file system is mounted, references to the file system name that is mounted refer to the root directory on the mounted file system.
2. A file system can be mounted at only one point.
3. The *file_system_name* can represent a BFS path name or an NFS path name:
   - To mount a BFS file system:
     - The *file_system_name* value must be a fully-qualified BFS path name. See "Understanding Byte File System (BFS) Path Name Syntax" on page 6.
     - The *parm* operand is not used.
   - To mount an NFS file system:
     - The *file_system_name* value must be a fully-qualified NFS path name. See "Understanding Network File System (NFS) Path Name Syntax" on page 9.
     - The *parm* operand is used to specify local NFS mount options. These options are mapped by the BPXYMNT macro. See "BPXYMNT — Map the File System Parameters for the mount Service" on page 435.
4. An NFS file system cannot be mounted as the root directory.

Example

The following code requests that file system /../VMBFS:BFS:USERS/ be mounted and readied for use. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structure, see "BPXYMTM — Map the Modes for the mount and umount Services" on page 445.

```
         XC    MTM(MTM#LENGTH),MTM
         MVI   MTM1,MTMRDWR          Mount mode - read-write
         MVC   BUFLENA,=F'2'         Max 1023
         MVC   BUFFERA(02),=CL02'/u'
         LA    R6,LFSTOMNT
         ST    R6,FSLEN
         MVC   FSNAME(LFSTOMNT),FSTOMNT
         MVC   FSTYPE(8),=CL08'VMBFS'
         SPACE ,
         CALL  BPX1MNT,              Ready a file system for use       +
               (BUFLENA,             Input: Mount point length         +
               BUFFERA,              Input: Mount point name           +
               FSLEN,                Input: File system name length    +
               FSNAME,               Input: File system name           +
               FSTYPE,               Input: File system type (8 char)  +
               MTM,                  Input: Mount mode         BPXYMTM +
               =A(0),                Input: Parm length, future        +
               =A(0),                Input: Parm, future               +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
⋮
FSTOMNT  DC    C'/../VMBFS:BFS:USERS/'
LFSTOMNT EQU   L'FSTOMNT
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBUSY	The file system to mount is quiesced, or no more locks are available. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JROutOfLocks, JRQuiesced.
ECMSERR	An internal error occurred.
ECMSSTORAGE	You have reached the maximum number of mounts (10) for your virtual machine.
EINVAL	Parameter error. Verify the *mount_mode* and *file_system_type* parameter values. Other reasons for this error include: the mount point is a root of a file system; the file system is already mounted; the *parm_length* is too long; *parm* contains invalid values; *parm* specified that NETRC DATA should be used, but the file was not found or did not contain username or password information for the remote host. The specified translation table (fn TCPXLBIN) was not found, or the default translation table, POSIX TCXLBIN, was not found. Consult the reason code to determine the exact reason the error occurred.
EIO	An I/O error occurred.
ELOOP	A loop exists in symbolic links encountered during resolution of the *file_system_name* argument. This error is issued if more than 8 symbolic links are detected in the resolution of the file system name.
ENOENT	The mount point does not exist. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRMountPt.
ENOTDIR	The mount point is not a directory. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRMountPt.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Service

Another callable service related to this service is:

- “umount (BPX1UMT) — Remove a Virtual File System” on page 375.

msgctl (BPX1QCT) — Perform Message Queue Control Operations

BPX1QCT

message_queue_ID
command
buffer_address
return_value
return_code
reason_code

Purpose

Use the msgctl (BPX1QCT) service to do various message queue control operations, including getting status, changing variables, and removing a message queue from the system.

Parameters

message_queue_ID
(input,INT,4) is a variable for specifying the message queue identifier. This value is returned by the msgget (BPX1QGT) service.

command
(input,INT,4) is a variable for specifying a command that identifies the operation to be performed. The command constants are defined in the BPXYIPCP macro. See "BPXYIPCP — Map Interprocess Communications Permissions" on page 431. The possible commands are:

Command
Operation

IPC_STAT
Obtains status information about *message_queue_ID*, if the current process has read permission. This information is stored in the area pointed to by the *buffer_address* parameter and mapped by the MSQID_DS data structure in the BPXYMSG macro.

IPC_SET
Sets the values of IPC_UID, IPC_GID, IPC_MODE, and MSG_QBYTES for *message_queue_ID*. The values to be set are taken from the MSQID_DS data structure pointed to by the *buffer_address* parameter. You can specify any values for IPC_UID and IPC_GID. For IPC_MODE, you can specify only the mode bits defined for the *message_flags* parameter of the msgget (BPX1QGT) service.

Note: The IPC_ values set with this command are defined in the BPXYIPCP macro and mapped into the MSG_PERM field of the MSQID_DS structure in the BPXYMSG macro. In addition, the IPC_MODE field in BPXYIPCP is mapped by the BPXYMODE macro.

IPC_RMID
Removes *message_queue_ID* from the system. This operation removes the identifier and destroys the message queue and the MSQID_DS data structure associated with it.

The IPC_SET and IPC_RMID operations can be performed only by a process that has either appropriate privileges or an effective user ID equal to the value of IPC_CUID or IPC_UID in the MSQID_DS data structure associated with *message_queue_ID*.

For the MSQID_DS data structure, see "BPXYMSG — Map Interprocess Communications Message Queues" on page 439.

buffer_address
(input,INT,4) is a variable for specifying the address of the buffer to be used for message queue information. The buffer is mapped by the MSQID_DS data structure in the BPXYMSG macro.

return_value
(output,INT,4) is a variable where the service returns 0 if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. Changing the access permissions affects only message queue service invocations that occur after msgctl (BPX1QCT) has returned. Both msgsnd (BPX1QSN) and msgrcv (BPX1QRC), which are waiting while the permission bits are changed by msgctl (BPX1QCT), are not affected.
2. The IPC_SET operation can change permissions, which may affect the ability of a thread to use the message queue callable services.
3. Quiescing a message queue will stop additional messages from being added, while allowing existing messages to be received. A message queue can be quiesced by using the IPC_SET command to clear write permission bits.
4. A message queue can also be quiesced by using the IPC_SET command to reduce MSG_QBYTES to zero. (Note that it would take a superuser to raise the limit again.) Requesters would receive an EAGAIN return code or would wait.
5. When an IPC_RMID command is processed, all waiting threads regain control with a return value of -1, a return code of EIDRM, and a reason code of JRIpcRemoved.
6. If you do not wish to change all the fields in the MSQID_DS data structure, first call the msgctl (BPX1QCT) service with the IPC_STAT command to initialize the buffer, then call the service again with the IPC_SET command to make your changes.
7. For an IPC_RMID operation, the removal of the message queue ID will be complete by the time control is returned to the caller.

Characteristics and Restrictions

The invoker is restricted by the ownership, read, and read-write permissions defined by the msgget (BPX1QGT) and msgctl (BPX1QCT) services.

Example

The following code removes a message queue from the system. For the data structure, see "BPXYMSG — Map Interprocess Communications Message Queues" on page 439.

```
         CALL  BPX1QCT,             Message queue control (msgctl)    +
               (MSG_ID,             Input: MessageQueueID             +
               =A(IPC_RMID),        Input: Action to take     BPXYIPCP+
               =A(0),               Input: ->MSGID_DS or 0    BPXYMSG +
               RETVAL,              Return value: 0, -1               +
               RETCODE,             Return code                       +
               RSNCODE),            Reason code                       +
               VL,MF=(E,PLIST)      ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The IPC_STAT command was specified, but the calling process does not have read permission. The following reason code can accompany this return code: JRIpcDenied.
EINVAL	One of the following conditions is true: • *message_queue_ID* is not a valid message queue identifier. • *command* is not a valid command. • The mode bits set by the IPC_SET command were not valid. The following reason codes can accompany this return code: JRIpcBadFlags, JRMsqQBytes, JRIpcBadID.
EPERM	One of the following conditions is true: • The IPC_SET or IPC_RMID command was specified, but the caller has neither appropriate privileges nor an effective user ID equal to the value of IPC_CUID or IPC_UID in the MSQID_DS data structure associated with *message_queue_ID*. • The IPC_SET command was specified, and an attempt was made to increase the MSG_QBYTES value, but the caller does not have appropriate privileges. The following reason codes can accompany this return code: JRIpcDenied, JRMsqQBytes.
EFAULT	The *buffer_address* parameter specified an address that caused the service to program check. The following reason code can accompany this return code: JRBadAddress.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “msgget (BPX1QGT) — Create or Find a Message Queue” on page 172
- “msgrcv (BPX1QRC) — Receive a Message from a Message Queue” on page 175
- “msgsnd (BPX1QSN) — Send a Message to a Message Queue” on page 178

msgget (BPX1QGT) — Create or Find a Message Queue

BPX1QGT

key
message_flags
return_value
return_code
reason_code

Purpose

Use the msgget (BPX1QGT) service to create a new message queue or find an existing message queue (if the user is allowed to access it). The service returns a system-assigned message queue identifier.

Parameters

key
(input,INT,4) is a variable for specifying a user-defined value that identifies a message queue. The *key* serves as a lookup value to determine if an associated message queue identifier already exists. If an associated message queue identifier does not already exist, the *key* value becomes associated with the message queue identifier created by this request.

The reserved key value IPC_PRIVATE may also be specified. IPC_PRIVATE is sometimes used when a process does not want to share a message queue or when it wants to privately control access to the message queue by other processes. The IPC_PRIVATE constant is defined in the BPXYIPCP macro. See “BPXYIPCP — Map Interprocess Communications Permissions” on page 431.

message_flags
(input,INT,4) is a variable for specifying the type of action to be performed and the permissions to be assigned. Valid values for this parameter include any combination of the following flags (additional bits will cause an EINVAL return code):

- These flags are defined in the BPXYIPCP macro and the values are mapped onto the S_TYPE field in the BPXYMODE macro:

  IPC_CREAT
  Creates a message queue if the specified *key* is not associated with a message queue identifier. IPC_CREAT is ignored when the IPC_PRIVATE reserved key is specified.

  IPC_EXCL
  Causes the service to fail if the specified *key* has an associated message queue identifier. IPC_EXCL is ignored when the IPC_PRIVATE reserved key is specified or the IPC_CREAT flag is not set.

- These flags are defined in the BPXYMODE macro and are a subset of the access permissions that apply to files:

  S_IRUSR
  Permits the process that owns the message queue to read it.

  S_IWUSR
  Permits the process that owns the message queue to alter it.

  S_IRGRP
  Permits the group associated with the message queue to read it.

  S_IWGRP
  Permits the group associated with the message queue to alter it.

S_IROTH
Permits others to read the message queue.

S_IWOTH
Permits others to alter the message queue.

See “BPXYIPCP — Map Interprocess Communications Permissions” on page 431 and “BPXYMODE — Map Mode Constants” on page 437.

return_value
(output,INT,4) is a variable where the service returns the message queue identifier associated with *key* if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. As long as a thread knows the message queue identifier and access is permitted, the thread can issue msgctl (BPX1QCT), msgsnd (BPX1QSN), or msgrcv (BPX1QRC) calls for that message queue, and msgget is not needed.
2. This service creates a data structure defined by MSQID_DS, if either of the following is true:
   - IPC_PRIVATE is specified in the *key* parameter.
   - The IPC_CREAT flag is set, and the specified *key* value does not already have a message queue identifier associated with it.

   The MSQID_DS data structure is defined in the BPXYMSG macro, and some values are mapped into it from the BPXYIPCP macro. See “BPXYMSG — Map Interprocess Communications Message Queues” on page 439 and “BPXYIPCP — Map Interprocess Communications Permissions” on page 431.
3. Upon creation, the MSQID_DS data structure is initialized as follows:
   - IPC_CUID and IPC_UID are set to the effective user ID of the calling task.
   - IPC_CGID and IPC_GID are set to the effective group ID of the calling task.
   - The low-order 9-bits of IPC_MODE are equal to the low-order 9-bits of the *message_flags* parameter.
   - MSG_QBYTES is set to the system limit defined by parmlib.
4. The message queue is removed from the system by calling the msgctl (BPX1QCT) service with the IPC_RMID command.
5. Users of message queues are responsible for removing them when they are no longer needed. Failure to do so will tie up system resources.

Characteristics and Restrictions

1. There is a maximum number of message queues allowed in the system.
2. The invoker is restricted by the ownership, read, and read-write permissions for the specified message queue as defined by the msgget (BPX1QGT) and msgctl (BPX1QCT) services.

Example

The following code creates a private message queue. For the data structure, see “BPXYMSG — Map Interprocess Communications Message Queues” on page 439.

```
         MVI   S_TYPE,IPC_CREAT+IPC_EXCL     Error if exists
         MVI   S_MODE1,0             Not used
         MVI   S_MODE2,S_IRUSR       All read and write permissions
         MVI   S_MODE3,S_IWUSR+S_IRGRP+S_IWGRP+S_IROTH+S_IWOTH
```

```
         SPACE ,
         CALL  BPX1QGT,              Create a message queue            +
               (=A(IPC_PRIVATE),     Input: Key                        +
               S_MODE,               Input: Creation flags BPXYMODE/IPC+
               RETVAL,               Return value: -1 or semaphore ID  +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
         SPACE ,
         ICM   R15,B'1111',RETVAL    Test return value
         BNP   PSEUDO                Branch on msgget failure
         ST    R15,MSG_ID            Store MSG_ID associated with key
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	A message queue identifier exists for the specified *key*, but access permission, as specified by the low-order 9-bits of the *message_flags* parameter (the S_ flags), is not granted. The following reason code can accompany this return code: JRIpcDenied.
EEXIST	A message queue identifier exists for the specified *key*, and the IPC_CREAT and IPC_EXCL flags are both set. The following reason code can accompany this return code: JRIpcExists.
EINVAL	The *message_flags* parameter included bits not supported by this service. The following reason code can accompany this return code: JRIpcBadFlags.
ENOENT	A message queue identifier does not exist for the specified *key*, and the IPC_CREAT flag was not set. The following reason code can accompany this return code: JRIpcNoExist.
ENOSPC	A message queue is to be created, but the system-imposed limit on the maximum number of message queue identifiers allocated system-wide would be exceeded. The following reason code can accompany this return code: JRIpcMaxIDs.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “msgctl (BPX1QCT) — Perform Message Queue Control Operations” on page 169
- “msgrcv (BPX1QRC) — Receive a Message from a Message Queue” on page 175
- “msgsnd (BPX1QSN) — Send a Message to a Message Queue” on page 178
- “w_getipc (BPX1GET) — Query Interprocess Communications” on page 391

msgrcv (BPX1QRC) — Receive a Message from a Message Queue

```
BPX1QRC
        message_queue_ID
        message_address
        message_ALET
        message_size
        message_type
        message_flag
        return_value
        return_code
        reason_code
```

Purpose

Use the msgrcv (BPX1QRC) service to receive a message from a message queue.

Parameters

message_queue_ID
(input,INT,4) is a variable for specifying the message queue identifier.

message_address
(input,INT,4) is a variable for specifying the address of a buffer mapped by the MSGBUF or MSGXBUF data structure in the BPXYMSG macro. See "BPXYMSG — Map Interprocess Communications Message Queues" on page 439.

message_ALET
(input,INT,4) is a variable for specifying the access list entry token (ALET) for *message_address* that identifies the address space or data space where the buffer resides:

- 0 indicates the buffer resides in the user's primary address space.
- All other values are ignored.

message_size
(input,INT,4) is a variable for specifying the length of the message text to be placed into the buffer pointed to by the *message_address* parameter.

If the MSG_INFO flag is set, the buffer should be 20 bytes longer than *message_size*. Otherwise, the buffer should be 4 bytes longer than *message_size*. It is the responsibility of the caller to ensure that the buffer is large enough to hold the message to be received.

The message may be truncated by setting the MSG_NOERROR flag. Specifying a *message_size* of 0 with the MSG_NOERROR flag is useful for receiving the message type without the message text.

message_type
(input,INT,4) is a variable for specifying the type of message to be received:

- If *message_type* is zero, the first message on the queue is received.
- If *message_type* is greater than zero, the first message of that message type is received.
- If *message_type* is less than zero, the first message of the lowest type that is less than or equal to the absolute value of *message_type* is received.

message_flag
(input,INT,4) is a variable for specifying receive options:

MSG_NOERROR
The received message is to be truncated to *message_size* (mapped in the BPXYMSG macro). The truncated part of the message is lost and no indication of the truncation is given to the caller.

MSG_INFO
The received message is to be of the MSGXBUF format mapped in the BPXYMSG macro, not the MSGBUF format.

IPC_NOWAIT
Indicates the action to be taken if a message of the desired type is not on the queue, as follows:

- If IPC_NOWAIT is specified, the caller will return immediately with an error (ENOMSG).
- If IPC_NOWAIT is not specified, the calling thread will suspend execution until one of the following occurs:
  - A message of the desired type is placed on the queue.
  - The message queue is removed from the system (EIDRM).
  - The caller receives a signal (EINTR).

The MSG_NOERROR and MSG_INFO flags are defined in the BPXYMSG macro. The IPC_NOWAIT flag is defined in the BPXYIPCP macro.

return_value
(output,INT,4) is a variable where the service returns the number of bytes of message text received (MSG_MTEXT) if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. Within the type specifications, the longest waiting thread will be reactivated first (FIFO). For example, if a message send for type 3 occurs when there are two threads waiting on message type 3 and one thread waiting on message type 2, the oldest waiter for message type 3 receive will be posted first.
2. Read access to the specified message queue is required.

Characteristics and Restrictions

The invoker is restricted by the ownership, read, and read-write permissions defined by the msgget (BPX1QGT) and msgctl (BPX1QCT) services.

Example

The following code receives a message from the message queue identified by MSG_ID. For the data structure, see "BPXYMSG — Map Interprocess Communications Message Queues" on page 439.

```
         LA    R15,BUFFERA           R15 -> Utility buffer
         ST    R15,BUFA
         USING MSGBUF,R15
         MVC   MSG_TYPE(4),=A(0)
         MVC   BUFLENA(4),=A(MSQ#LENGTH)
         MVC   FLAGS(4),=A(0)        Wait for message
         DROP  R15
         SPACE ,
         CALL  BPX1QRC,              Receive a message (msgrcv)        +
               (MSG_ID,              Input: MessageQueueID             +
               BUFA,                 Input: ->MSGBUF           BPXYMSG +
               PRIMARYALET,          Input: ALET of message buffer     +
               BUFLENA,              Input: Length MSGBUF              +
               =A(0),                Input: Message Type       BPXYMSG +
               FLAGS,                Input: Flags              BPXYIPCP+
```

```
                RETVAL,              Return value: 0, -1              +
                RETCODE,             Return code                      +
                RSNCODE),            Reason code                      +
                VL,MF=(E,PLIST)      ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
E2BIG	The length of the message text (MSG_MTEXT) is greater than the specified *message_size*, and the MSG_NOERROR flag is not set. The following reason code can accompany this return code: JRMsq2Big.
EACCES	Operation permission is denied to the calling task. The following reason code can accompany this return code: JRIpcDenied.
EIDRM	*message_queue_ID* was removed from the system while the invoker was waiting. The following reason code can accompany this return code: JRIpcRemoved.
EINTR	The function was interrupted by a signal. The following reason code can accompany this return code: JRIpcSignaled.
EINVAL	*message_queue_ID* is not a valid message queue identifier, or *message_size* is less than 0. The following reason codes can accompany this return code: JRIpcBadID, JRMsqBadSize.
EFAULT	The *message_address* parameter specified an address that caused the service to program check. The following reason code can accompany this return code: JRBadAddress.
ENOMSG	The queue does not contain a message of the desired type, and the IPC_NOWAIT flag is set. The following reason codes can accompany this return code: JRMsqNoMsg.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “msgctl (BPX1QCT) — Perform Message Queue Control Operations” on page 169
- “msgget (BPX1QGT) — Create or Find a Message Queue” on page 172
- “msgsnd (BPX1QSN) — Send a Message to a Message Queue” on page 178

msgsnd (BPX1QSN) — Send a Message to a Message Queue

BPX1QSN

message_queue_ID
message_address
message_ALET
message_size
message_flag
return_value
return_code
reason_code

Purpose

Use the msgsnd (BPX1QSN) service to send a message to a message queue.

Parameters

message_queue_ID
(input,INT,4) is a variable for specifying the message queue identifier.

message_address
(input,INT,4) is a variable for specifying the address of a buffer that contains the message to be sent. The buffer is mapped by the MSGBUF data structure in the BPXYMSG macro. See "BPXYMSG — Map Interprocess Communications Message Queues" on page 439. The message type (MSG_TYPE field) is the first word of the message and must be greater than zero.

message_ALET
(input,INT,4) is a variable for specifying the access list entry token (ALET) for *message_address* that identifies the address space or data space where the buffer resides:

- O indicates the buffer resides in the user's primary address space.
- All other values are ignored.

message_size
(input,INT,4) is a variable for specifying the length of the message text contained in the buffer pointed to by the *message_address* parameter. This length does not include the 4-byte MSG_TYPE field that precedes the message text field (MSG_MTEXT). For example, a message with a MSG_TYPE and no MSG_MTEXT would have a *message_size* of 0.

message_flag
(input,INT,4) is a variable that specifies the action to be taken if one or more of the following conditions are true:

- Placing the message on the message queue would cause the current number of bytes on the message queue (*msg_cbytes*) to be greater than the maximum number of bytes allowed on the message queue (*msg_qbytes*).
- The total number of messages on the message queue (*msg_qnum*) is equal to the system-imposed limit.

The actions are as follows:

- If IPC_NOWAIT is specified, the caller will return immediately with an error (EAGAIN).
- If IPC_NOWAIT is not specified, the calling thread will suspend execution until one of the following occurs:

- The message is sent.
- The message queue is removed from the system (EIDRM).
- The caller receives a signal (EINTR).

The IPC_NOWAIT flag is defined in the BPXYIPCP macro.

return_value
(output,INT,4) is a variable where the service returns a value of 0 if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Note

Write access to the specified message queue is required.

Characteristics and Restrictions

The invoker is restricted by the ownership, read, and read-write permissions defined by the msgget (BPX1MGT) and msgctl (BPX1MCT) services.

Example

The following code adds a message to the message queue identified by MSG_ID. For the data structure, see "BPXYMSG — Map Interprocess Communications Message Queues" on page 439.

```
         LA    R15,BUFFERA           R15 -> Utility buffer
         ST    R15,BUFA
         USING MSGBUF,R15
         MVC   MSG_TYPE(4),=A(0)
         MVC   MSG_MTEXT(11),=CL11'QSN MSG TEXT'
         MVC   BUFLENA(4),=A(15)
         MVC   FLAGS(4),=A(IPC_NOWAIT)   Don't wait on queue full
         DROP  R15
         SPACE ,
         CALL  BPX1QSN,              Send a message (msgsnd)           +
               (MSG_ID,              Input: MessageQueueID             +
               BUFA,                 Input: ->MSGBUF           BPXYMSG +
               PRIMARYALET,          Input: ALET of message buffer     +
               BUFLENA,              Input: Length MSGBUF              +
               FLAGS,                Input: Flags              BPXYIPCP+
               RETVAL,               Return value: 0, -1               +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	Operation permission is denied to the calling task. The following reason code can accompany this return code: JRIpcDenied.
EAGAIN	The message cannot be sent, and *message_flag* is set to IPC_NOWAIT. The following reason codes can accompany this return code: JRMsqQueueFullMessages, JRMsqQueueFullBytes.

Return Code	Explanation
EIDRM	*message_queue_ID* was removed from the system while the invoker was waiting. The following reason code can accompany this return code: JRIpcRemoved.
EINTR	The function was interrupted by a signal, and the message was not sent. The following reason code can accompany this return code: JRIpcSignaled.
EINVAL	One of the following conditions is true: • *message_queue_ID* is not a valid message queue identifier. • The MSG_TYPE field of the message is less than 1. • *message_size* is less than zero or greater than the system-imposed limit. The following reason codes can accompany this return code: JRIpcBadID, JRMsqBadSize, JRMsqBadType.
EFAULT	The *message_address* parameter specified an address that caused the service to program check. The following reason code can accompany this return code: JRBadAddress.
ENOMEM	There are not enough system storage exits to send the message, and the message was not sent. The following reason code can accompany this return code: JRSmNoStorage.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “msgctl (BPX1QCT) — Perform Message Queue Control Operations” on page 169
- “msgget (BPX1QGT) — Create or Find a Message Queue” on page 172
- “msgrcv (BPX1QRC) — Receive a Message from a Message Queue” on page 175

open (BPX1OPN) — Open a File

BPX1OPN

pathname_length
pathname
options
mode
return_value
return_code
reason_code

Purpose

Use the open (BPX1OPN) service to gain access to a file and create a file descriptor for it. You identify the file by its path name.

Parameters

pathname_length
(input,INT,4) is a variable for specifying the length of the *pathname* parameter.

pathname
(input,CHAR,*pathname_length*) is a variable for specifying the path name of the file to be opened. See "Understanding Byte File System (BFS) Path Name Syntax" on page 6.

options
(input,INT,4) is a variable for specifying the binary flags that describe how the file is to be opened. For descriptions of the options, see "Usage Notes" on page 181.

This parameter is mapped by the BPXYOPNF macro. See "BPXYOPNF — Map Flag Values for the open and fcntl Services" on page 447.

mode
(input,INT,4) is a variable for specifying the permissions the caller grants to itself, to its groups, and to any user. This parameter is mapped by the BPXYMODE macro. See "BPXYMODE — Map Mode Constants" on page 437.

If create or exclusive create is not specified on the options parameter, the mode parameter is ignored.

return_value
(output,INT,4) is a variable where the service stores the file descriptor if the file was opened successfully, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

When a file is created with the Create or Exclusive_create options of the *Options* parameter, the file permission bits as specified in the *Mode* parameter are modified by the process's file creation mask (see "umask (BPX1UMK) — Set or Return the File Mode Creation Mask" on page 374) and then used to set the file permission bits of the file being created.

Exclusive Create Option: If the exclusive create bit is set and the create bit is not set, the exclusive create bit is ignored.

Truncate Option: Turning on the truncate bit opens the file as though it had been created earlier, but never written into. The mode and owner of the file do not change (although the change time and modification time do); but the file's contents are discarded. The file offset, which indicates where the next write is to occur, points to the first byte of the file.

Nonblock Option: A FIFO special file is a shared file from which the first data written is the first data read. The Nonblock option is a way of coordinating write and read requests between processes sharing a FIFO special file. It works this way, provided that no other conditions interfere with opening the file successfully:

- If a file is opened read-only and Nonblock is specified, the open request succeeds. Control returns to the caller immediately.
- If a file is opened write-only and Nonblock is specified, the open request completes successfully, provided that another process has the file open for reading. If another process does not have the file open for reading, the request ends with *return_value* set to -1.
- If a file is opened read-only and Nonblock is omitted, the request is blocked (control is not returned to the caller) until another process opens the file for writing.
- If a file is opened write-only and Nonblock is omitted, the request is blocked (control is not returned to the caller) until another process opens the file for reading.

Example

The following code opens file **usr/inv/nov.d** with user read-write, group read and other read. A file descriptor (FILEDESC) is returned. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure see “BPXYOPNF — Map Flag Values for the open and fcntl Services” on page 447 and “BPXYMODE — Map Mode Constants” on page 437.

```
         MVC   BUFFERA(13),=CL13'usr/inv/nov.d'
         MVC   BUFLENA,=F'13'
         XC    S_MODE,S_MODE
         MVI   S_MODE2,S_IRUSR       User read/write, group read,
         MVI   S_MODE3,S_IWUSR+S_IRGRP+S_IROTH           other read
         XC    O_FLAGS(OPNF#LENGTH),O_FLAGS
         MVI   O_FLAGS4,O_CREAT+O_RDWR Create, open for read and write
         SPACE ,
         CALL  BPX1OPN,              Open a file                       +
               (BUFLENA,             Input: Pathname length            +
               BUFFERA,              Input: Pathname                   +
               O_FLAGS,              Input: Access            BPXYOPNF +
               S_MODE,               Input: Mode              BPXYMODE +
               RETVAL,               Return value:-1 or file descriptor+
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ---------------------------------
         ICM   R15,B'1111',RETVAL    Test RETVAL
         BL    PSEUDO                Branch if negative (-1 = failure)
         ST    R15,FILEDESC          Store the file descriptor
```

VM-Related Information

The Execution access requested bit is used by the exec service (see “exec (BPX1EXC) — Run a Program” on page 72) to verify that the process has permission to run the specified file. When open succeeds, the specified file is treated as read-only for this case.

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	Reasons for being denied access include these: • The calling process does not have permission to search one of the directories specified in the *pathname* parameter. • The calling process does not have permission to open the file in the way specified by *options* parameter. • The file does not exist, and the calling process does not have permission to write into files in the directory the file would have been created in. • The truncate option was specified, but the process does not have write permission for the file.
EAGAIN	Resources were temporarily unavailable.
EBUSY	Typical causes: • An attempt was made to open a terminal which is already in use by another process. • The process has already opened one terminal. Consult the reason code to determine the exact reason the error occurred.
EEXIST	The exclusive create option was specified, but the file already exists. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRFileExistsExclFlagSet.
EINTR	The open operation was interrupted by a signal.
EINVAL	The *options* parameter does not specify a valid combination of the O_RDONLY, O_WRONLY and O_TRUNC bits, or the file type specified in the *mode* parameter is not valid. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRInvOpenFlags and JROpenFlagConflict.
EISDIR	The file specified by *pathname* is a directory and the *options* parameter specifies write or read/write access. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRDirWriteRequest.
ELOOP	A loop exists in symbolic links encountered during resolution of the *pathname* argument. This error is issued if more than 8 symbolic links are detected in the resolution of the path name.
EMFILE	The process has reached the maximum number of file descriptors it can have open.
ENAMETOOLONG	The *pathname* parameter is longer than 1023 characters, or a component of the path name is longer than 255 characters. (CMS does not support file name truncation.)
ENFILE	CMS has reached the maximum number of file descriptors it can have open.

Return Code	Explanation
ENODEV	Typical causes: • An attempt was made to open a character special file for a device not supported by CMS. • An attempt was made to open a character special file for a device which is not yet initialized. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRNoCTTY.
ENOENT	Typical causes: • The request did not specify that the file was to be created, but the file named by *pathname* was not found. • The request asked for the file to be created, but some component of *pathname* was not found, or the *pathname* parameter was blank. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JREndingSlashOCreat, JRNoFileNoCreatFlag, and JRQuiescing.
ENOSPC	The directory or file system intended to hold a new file has insufficient space.
ENOTDIR	A component of *pathname* is not a directory.
ENXIO	The open request specified write-only and nonblock for a FIFO special file, but no process has the file open for reading. For terminals, it can mean that the major number associated with the path name is not valid.
EROFS	The *pathname* parameter names a file on a read-only file system, but options that would allow the file to be altered were specified: write-only, read/write, truncate, or—for a new file—create. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRReadOnlyFileSetWriteReq and JRReadOnlyFileSetCreatReq.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

- “close (BPX1CLO) — Close a File or Socket” on page 34
- “exec (BPX1EXC) — Run a Program” on page 72
- “fcntl (BPX1FCT) — Control Open File Descriptors” on page 88
- “lseek (BPX1LSK) — Change the File Offset” on page 154
- “read (BPX1RED) — Read from a File or Socket” on page 228
- “stat (BPX1STA) -- Get Status Information about a File by Path Name” on page 340
- “write (BPX1WRT) — Write to a File or Socket” on page 401
- “umask (BPX1UMK) — Set or Return the File Mode Creation Mask” on page 374.

opendir (BPX1OPD) — Open a Directory

BPX1OPD

directory_name_length
directory_name
return_value
return_code
reason_code

Purpose

Use the opendir (BPX1OPD) service to open a directory so that it can be read with the readdir (BPX1RDD) service.

Parameters

directory_name_length
(input,INT,4) is a variable for specifying the length of the *directory_name* parameter.

directory_name
(input,CHAR,*directory_name_length*) is a variable for specifying the name of the directory to be opened. Each component (subdirectory) of the directory name can be up to 255 characters. The complete directory name can be up to 1023 characters and does not require an ending null character.

return_value
(output,INT,4) is a variable where the service stores a directory file descriptor describing the specified directory, if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason codes is returned only if *return_value* is -1.

Usage Notes

1. The opendir (BPX1OPD) service opens a directory so that the first readdir (BPX1RDD) service call starts reading at the first entry in the directory.
2. The *return_value* parameter contains a file descriptor for a directory only. It can be used only as input to services that expect a directory file descriptor. These services are closedir (BPX1CLD), rewinddir (BPX1RWD), and readdir (BPX1RDD).

Example

The following code opens directory **/etc/passwd** so that it can be read by readdir. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   BUFLENA,=F'11'
         MVC   BUFFERA(11),=CL11'/etc/passwd'
         SPACE ,
         CALL  BPX1OPD,              Open a directory                 +
               (BUFLENA,             Input: Directory name length     +
               BUFFERA,              Input: Directory name            +
               RETVAL,               Return value:-1 or directory f.d. +
               RETCODE,              Return code                      +
```

```
         RSNCODE),             Reason code                        +
         VL,MF=(E,PLIST)       ---------------------------------
ICM      R15,B'1111',RETVAL    Test RETVAL
BL       PSEUDO                Branch if negative (-1 = failure)
ST       R15,DIRECTDES         Store the directory descriptor
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The process does not have permission to search some component of the name specified as *directory_name*, or does not have permission to work with the directory itself.
ELOOP	A loop exists in symbolic links encountered during resolution of the *directory_name* argument. This error is issued if more than 8 symbolic links are detected in the resolution of the directory name.
EMFILE	Too many other files are already open for the process.
ENAMETOOLONG	The *directory_name* parameter is longer than 1023 bytes, or a component of the directory name is longer than 255 bytes.
ENFILE	Too many files are already open in CMS.
ENOENT	The specified directory was not found. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JROpenDirNotFound and JRQuiescing.
ENOTDIR	Some component of the directory name is not a directory. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRTargetNotDir.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “closedir (BPX1CLD) — Close a Directory” on page 36
- “readdir (BPX1RDD) — Read an Entry from a Directory” on page 231
- “rewinddir (BPX1RWD) — Reposition a Directory Stream to the Beginning” on page 254.

openvmf (BPX1VM5) — Perform OpenExtensions Platform Functions

BPX1VM5

function_code
buffer_length
buffer
return_value
return_code
reason_code

Purpose

Use the openvmf (BPX1VM5) service to perform functions specific to the OpenExtensions platform.

Parameters

function_code

(input,INT,4) is a variable for specifying the function to be performed. This variable is mapped by the BPXYVM5 macro. See "BPXYVM5 — Map Function Code Values for the openvmf Service" on page 482. The possible function codes are:

Function Code	Meaning
VM5_RELEASE_TOKEN	Directs the Byte File System (BFS) to release any BFS file tokens that may have been manipulated by the caller.
VM5_FILEPOOL_ADMIN_RESPECT	Directs BFS to *respect* file pool administration authority when determining whether a file can be accessed. BFS file access is based on the user's UID and GID values, but if the user is a file pool administrator, the user is also given the additional privileges given to a file pool administrator.
VM5_FILEPOOL_ADMIN_IGNORE	Directs BFS to *ignore* file pool administration authority when determining whether a file can be accessed. BFS file access is based only on the user's UID and GID values.
VM5_RESOLVE_INO	Resolves an INO into a fully-qualified BFS path name. **Note:** This function is intended for IBM use only.
VM5_RESOLVE_PATH	Resolves a partially- or fully-qualified BFS path name, which may contain symbolic links or mount external links, to its fully-qualified BFS system root (FQR). The FQR takes the form, `/.../VMBFS:`*`filepoolid`*`:`*`filespaceid`*.

Function Code	Meaning
VM5_SET_SGID	Receives a pointer to an array of supplementary group IDs (sGIDs) and resets the sGID list.
VM5_SET_ALL_IDS	Receives a pointer to an array that contains supplementary group IDs (sGIDs), an effective GID (eGID), and an effective UID (eUID) and then resets the eGID, resets the sGID list, and resets the eUID, in that order.
VM5_GET_FILESYS_TYPE	Obtains the file system type for a given path name.

buffer_length
(input/output,INT,4) is a variable for specifying the length of the *buffer* parameter. For some functions, the service may return a value. See the Usage Notes.

buffer
(input/output,CHAR,*buffer_length*) is a variable to provide information that is dependent upon the function code specified. For some functions, the service may return a value. See the Usage Notes.

return_value
(output,INT,4) is a variable where the service returns `0` if the request completes successfully, or `-1` if the request is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. The buffer is not used for function codes VM5_RELEASE_TOKEN, VM5_FILEPOOL_ADMIN_RESPECT, and VM5_FILEPOOL_ADMIN_IGNORE. In these cases, the buffer length and buffer parameters are ignored.
2. OpenExtensions services make use of CMS Multitasking services. An application that uses OpenExtensions services cannot issue OpenExtensions calls from interrupt handlers and cannot use non-CMS Multitasking wait services. However, if these conditions cannot be met, then some problems can be avoided by calling openvmf (BPX1VM5) with function code VM5_RELEASE_TOKEN to release BFS file tokens immediately before entering any kind of non-CMS Multitasking wait state.
3. For function code VM5_RESOLVE_INO, the buffer length and buffer parameters are used as follows:
   - **Input**
     - *buffer_length* must be greater than or equal to 1200 bytes. This is PATH_MAX (1023 bytes) plus null plus the maximum system root length (FQR = 27 bytes) plus the BPXYSTAT length.
     - *buffer* must contain the following values:

       file pool ID (CHAR,8)
       file space ID (CHAR,8)
       INO (INT,4)
   - **Output**
     - *buffer_length* contains the length of the BFS path name.
     - *buffer* contains the attributes of the BFS object followed by the fully-qualified BFS path name.

       The caller can tell where the path name starts because the caller knows the length of the attribute section (it is mapped by BPXYSTAT). The caller can tell whether attributes are being returned (they

were not returned prior to VM/ESA® 2.4.0) by the very last byte of the buffer. If the last byte is X'FF', then attributes are being returned; otherwise, the last byte is 0.

In resolving the INO, this function does not traverse mount points or resolve symbolic links.

4. For function code VM5_RESOLVE_PATH, the buffer length and buffer parameters are used as follows:
   - **Input**
     - *buffer_length* must be greater than or equal to 1024 bytes (PATH_MAX + null).
     - *buffer* must contain a null-terminated BFS path name.
   - **Output**
     - *buffer_length* contains the length of the fully-qualified BFS system root (FQR).
     - *buffer* contains the FQR. The FQR may include an ending slash and is null-terminated.
5. For function code VM5_SET_SGID, the buffer length and buffer parameters are used as follows:
   - **Input**
     - *buffer_length* must be greater than or equal to the length of the buffer contents described below.
     - *buffer* contains the sGID count (first 4 bytes) and a pointer to a list of sGIDs (second 4 bytes).

       If the sGID count is zero, the sGID pointer is ignored and the sGID list for the active process is cleared.
   - **Output**
     - None.
6. For function code VM5_SET_ALL_IDS, the buffer length and buffer parameters are used as follows:
   - **Input**
     - *buffer_length* must be greater than or equal to the length of the buffer contents described below.
     - *buffer* must contain the following fields:

       eUID (INT,4)
       eGID (INT,4)
       sGID count (INT,4)
       pointer to an sGID list (INT,4)
       failed-call word (INT,4)

       If the sGID count is zero, the sGID pointer is ignored and the sGID list for the active process is cleared. The failed-call word is an output field.
   - **Output**
     - If a failure occurs, *buffer* contains a value in the failed-call word that indicates which of the subfunctions failed:

Word
Failure

1
Setting the sGID list

2
Setting the eGID

3
Setting the eUID

7. For function code VM5_GET_FILESYS_TYPE, the buffer length and buffer parameters are used as follows:
   - **Input**
     - *buffer_length* is the length of the buffer, which is a minimum of 8, or the length of the null-terminated BFS path name provided in *buffer*.
     - *buffer* must contain a null-terminated BFS path name.
   - **Output**
     - *buffer* contains the file system type as defined by BPXYVM5. For example, if the path name represents a file residing in an NFS-mounted file system, BPXFSNFS is returned.

Example

The following code forces BFS to release any file tokens that may be held by the application. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYVM5 — Map Function Code Values for the openvmf Service” on page 482.

```
         MVC   BUFLENA,=F'0'
         LA    R15,VM5_RELEASE_TOKEN
         ST    R15,VMFUNC
         SPACE ,
         CALL  BPX1VM5,              Perform z/VM Platform function  +
               (VMFUNC,              Input: openvmf, BPXYVM5           +
               BUFLENA,              Input: Pathname length            +
               BUFFERA,              Input: Pathname                   +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

The following code forces BFS to ignore file pool administration authority when determining whether a file can be accessed.

```
         MVC   BUFLENA,=F'0'
         LA    R15,VM5_FILEPOOL_ADMIN_IGNORE
         ST    R15,VMFUNC
         SPACE ,
         CALL  BPX1VM5,              Perform z/VM Platform function  +
               (VMFUNC,              Input: openvmf, BPXYVM5           +
               BUFLENA,              Input: Pathname length            +
               BUFFERA,              Input: Pathname                   +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EINVAL	The *function_code*, *buffer*, or *buffer_length* parameter is incorrect.
ENOENT	The BFS object does not exist.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

openvmf7 (BPX1VM7) — Perform z/VM NFS Client Functions

BPX1VM7

function_code
foreign_host_length
foreign_host
buffer_length
buffer
return_value
return_code
reason_code

Purpose

Use the openvmf7 (BPX1VM7) service to perform functions specific to the Network File System (NFS) Client for the z/VM platform.

Parameters

function_code
(input,INT,4) is a variable for specifying the function to be performed. This variable is mapped by the BPXYVM7 macro. See "BPXYVM7 — Map the Function Code Values and Buffer for the openvmf7 Service" on page 484. The possible function codes are:

Function Code	Meaning
VM7_GET_EXPORT_LIST	Obtain the list of file systems exported by *foreign_host*, and the list of clients allowed to mount each one.
VM7_GET_DUMP_LIST	Obtain the list of file systems mounted at *foreign_host*.
VM7_PCNFS_AUTH	Authenticate user ID at *foreign_host*, and retrieve UID and GID information in effect there.

foreign_host_length
(input,INT,4) is a variable specifying the length of the *foreign_host* parameter.

foreign_host
(input,CHAR,*foreign_host_length*) is a variable specifying the name of the remote host.

buffer_length
(input,INT,4) is a variable for specifying the length of the buffer parameter. See the Usage Notes.

buffer
(input/output,CHAR,buffer_length) is a variable for providing information that is dependent upon the function code specified. For some functions, the service may return a value. See the Usage Notes. The buffer is mapped by the BPXYVM7 macro. See "BPXYVM7 — Map the Function Code Values and Buffer for the openvmf7 Service" on page 484.

return_value
(output,INT,4) is a variable where the service returns `0` if the request completes successfully, or `-1` if the request is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. The buffer is used for all function codes.
2. For function code VM7_GET_EXPORT_LIST, the buffer length and buffer parameters are used as follows:
   - **Input**
     - *buffer_length* must be 4 bytes or greater.
   - **Output**
     - *buffer* contains the number of entries (VM7E_ENTRY_COUNT) in the first word and as much of the remaining information as will fit. See the mapping for VM7E_EXPORT_LIST in the BPXYVM7 macro.
3. For function code VM7_GET_DUMP_LIST, the buffer length and buffer parameters are used as follows:
   - **Input**
     - *buffer_length* must be 4 bytes or greater.
   - **Output**
     - *buffer* contains the number of entries (VM7D_ENTRY_COUNT) in the first word and as much of the remaining information as will fit. See the mapping for VM7D_DUMP_LIST in the BPXYVM7 macro.
4. For function code VM7_PCNFS_AUTH, the buffer length and buffer parameters are used as follows:
   - **Input**
     - *buffer_length* contains the size of the input buffer. A minimum size of VM7P_OUTPUT_LENGTH bytes is required.
     - *buffer* must contain the user name and password information. See the mapping for VM7P_PCNFS_INPUT in the BPXYVM7 macro.
   - **Output**
     - *buffer* contains the UID and GID information. See the mapping for VM7P_PCNFS_OUTPUT in the BPXYVM7 macro.

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EINVAL	The *function_code* parameter is incorrect. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code when the STANDARD TCPXLBIN file is not available: JRNFSCMntTCPXLBIN

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

pathconf (BPX1PCF) — Determine Configurable Path Name Variables Using Path Name

BPX1PCF

pathname_length
pathname
name
return_value
return_code
reason_code

Purpose

Use the pathconf (BPX1PCF) service to determine the current value of a configurable limit or option (variable) associated with a file or directory identified by its path name.

For the corresponding service using a file descriptor, see "fpathconf (BPX1FPC) — Determine Configurable Path Name Variables Using a Descriptor" on page 99.

Parameters

pathname_length
(input,INT,4) is a variable for specifying the length of the *pathname* parameter.

pathname
(input,CHAR,*pathname_length*) is a variable for specifying the path name of the file. See "Understanding Byte File System (BFS) Path Name Syntax" on page 6.

name
(input,INT,4) is a variable for specifying the path name variable to be returned. Use the BPXYPCF macro. See "BPXYPCF — Map Command Values for the pathconf and fpathconf Services" on page 448. The path name variables you can specify are:

Path Name Variable	Description
PC_CHOWN_RESTRICTED	The change ownership service, chown (BPX1CHO), is restricted to a process with appropriate privileges, and to changing the group ID (GID) of a file only to the effective group ID of the process or to one of its supplementary group IDs.
PC_LINK_MAX	Maximum value of a file's link count.
PC_MAX_CANON	Maximum number of bytes in a terminal canonical input line.
PC_MAX_INPUT	Minimum number of bytes for which space will be available in a terminal input queue; therefore, the maximum number of bytes a portable application may require to be typed as input before reading them.
PC_NAME_MAX	Maximum number of bytes in a file name (not a string length; count excludes a terminating null).
PC_NO_TRUNC	Path name components longer than 255 bytes generate an error.
PC_PATH_MAX	Maximum number of bytes in a path name (not a string length; count excludes a terminating null).

Path Name Variable	Description
PC_PIPE_BUF	Maximum number of bytes that can be written atomically when writing to a pipe.
PC_VDISABLE	Terminal special characters maintained by the system can be disabled using this character value. For information on querying and setting these special characters, see "tcgetattr (BPX1TGA) — Get the Attributes for a Terminal" on page 358 or "tcsetattr (BPX1TSA) — Set the Attributes for a Terminal" on page 365.

return_value
(output,INT,4) is a variable where the service returns the current value of the path name variable specified in the *name* parameter, or -1 if the request is not successful.

If the path name variable is PC_CHOWN_RESTRICTED and this option is active, the return value is set to 1. If this option is not active, the return value is set to 0.

If the path name variable is PC_NO_TRUNC and this option is active, the return value is set to 1. If this option is not active, the return value is set to 0.

If the path name variable does not have a limit for the specified file, the return value is set to -1 and the return code and reason code remain unchanged.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

If the path name variable does not have a limit for the specified file, the return value is set to -1 and the return code is unchanged.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

If the path name variable does not have a limit for the specified file, the return value is set to -1 and the reason code is unchanged.

Usage Notes

1. If *name* is PC_MAX_CANON, PC_MAX_INPUT, or PC_VDISABLE, and *pathname* does not refer to a terminal file, the service returns return value -1 and return code EINVAL.
2. If *name* is PC_NAME_MAX, PC_PATH_MAX, or PC_NO_TRUNC, and *pathname* does not refer to a directory, the service still returns the requested information using the parent directory of the specified file.
3. If *name* is PC_PIPE_BUF:
   - If *pathname* refers to a pipe or a FIFO, the return value applies to the referred-to object.
   - If *pathname* refers to a directory, the return value applies to any FIFOs that exist or can be created within the directory.
   - If *pathname* refers to any other type of file, the service returns return value -1 and return code EINVAL.
4. If *name* is PC_LINK_MAX and *pathname* refers to a directory, the return value applies to the directory.

Example

The following code extracts the current value for the configurable maximum number of bytes in a file name associated with **/usr/inv/network.t**. This example follows the rules of reentrancy. For linkage

information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYPCF — Map Command Values for the pathconf and fpathconf Services” on page 448.

```
         MVC   BUFFERA(18),=CL18'/usr/inv/network.t'
         MVC   BUFLENA,=F'18'
         SPACE ,
         CALL  BPX1PCF,              Get configurable pathname variable+
               (BUFLENA,             Input: Pathname length            +
               BUFFERA,              Input: Pathname                   +
               =A(PC_NAME_MAX),      Input: Options            BPXYPCF +
               RETVAL,               Return value: 0, -1 or variable   +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return one of the following return codes:

Return Code	Explanation
EACCES	Search permission is denied for a component of the path name.
EINVAL	Refer to the Usage Notes for situations where this return code is returned. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRNotSupportedForFileType.
ELOOP	A loop exists in symbolic links encountered during resolution of the *pathname* argument. This error is issued if more than 8 symbolic links are detected in the resolution of the path name.
ENAMETOOLONG	The *pathname* argument is longer than 1023 characters, or some component of the path name is longer than 255 characters. CMS does not support name truncation.
ENOENT	The named file does not exist, or the *pathname* argument points to an empty string. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRFileNotThere.
ENOTDIR	A component of the path name is not a directory.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Service

Another callable service related to this service is:

- “fpathconf (BPX1FPC) — Determine Configurable Path Name Variables Using a Descriptor” on page 99.

pause (BPX1PAS) — Suspend a Process Pending a Signal

BPX1PAS

return_value
return_code
reason_code

Purpose

Use the pause (BPX1PAS) service to suspend execution of the calling thread until delivery of a signal whose action is either to execute a signal-catching function or to end the thread.

Parameters

return_value
(output,INT,4) is a variable where the service returns -1 if completion of a signal-handling function causes control to be returned. The service does not otherwise return to the caller.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. A thread that calls pause (BPX1PAS) does not resume processing until a signal is delivered with an action to either process a signal-handling function or to end the thread. Some signals can be blocked by the thread's signal mask. See “sigprocmask (BPX1SPM) — Examine or Change a Thread's Signal Mask” on page 321 for details.
2. If an incoming unblocked signal ends the thread, pause (BPX1PAS) never returns to the caller.
3. If the signal action is to process a signal-catching function, the signal interface routine (SIR), defined by the cmssigsetup call, is given control when the pause (BPX1PAS) service returns.
4. A return code is set when any failures are encountered that prevent this function from completing successfully.

Characteristics and Restrictions

See Appendix E, “The Relationship of OpenExtensions Signals to Callable Services,” on page 557.

Example

The following code suspends execution of the invoker's thread until a signal is delivered. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         CALL  BPX1PAS,              Suspend execution                +
               (RETVAL,              Return value: -1 or not return   +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECMSERR	A CMS environmental or internal error has occurred. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRWrongSsave.
EINTR	A signal was received and handled successfully.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “alarm (BPX1ALR) — Set an Alarm” on page 18
- “kill (BPX1KIL) — Send a Signal to a Process” on page 146
- “sigaction (BPX1SIA) — Examine or Change a Signal Action” on page 315
- “sigprocmask (BPX1SPM) — Examine or Change a Thread's Signal Mask” on page 321
- “sigsuspend (BPX1SSU) — Change the Signal Mask and Suspend the Thread Until a Signal Is Delivered” on page 324
- “wait (BPX1WAT) — Wait for a Child Process to End” on page 385.

pipe (BPX1PIP) — Create an Unnamed Pipe

BPX1PIP

read_file_descriptor
write_file_descriptor
return_value
return_code
reason_code

Purpose

Use the pipe (BPX1PIP) service to create a pipe. A pipe is an I/O channel that a process can use to communicate with another process, another thread (in this same process or another process), or in some cases with itself. Data can be written into one end of the pipe and read from the other.

Parameters

read_file_descriptor
(output,INT,4) is a variable where the service stores the file descriptor for the read end of the pipe if the pipe is created successfully.

write_file_descriptor
(output,INT,4) is a variable where the service stores the file descriptor for the write end of the pipe if the pipe is created successfully.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. Processes can read from the *read_file_descriptor* and write to the *write_file_descriptor*. Data written will be read first-in, first-out (FIFO).
2. When the pipe (BPX1PIP) call creates a pipe, the O_NONBLOCK and FD_CLOEXEC flags are turned off on both ends of the pipe. You can turn these flags on with the fcntl (BPX1FCT) call. See "fcntl (BPX1FCT) — Control Open File Descriptors" on page 88.

Example

The following code creates a pipe. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551.

```
         CALL  BPX1PIP,              Create a pipe                    +
               (READFD,              Output: Read file descriptor     +
               WRITEFD,              Output: Write file descriptor    +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EMFILE	Opening the pipe would exceed the limit on the number of file descriptors that the process may have open.
ENFILE	Opening the pipe would exceed the number of files that the system can have open simultaneously.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “fcntl (BPX1FCT) — Control Open File Descriptors” on page 88
- “open (BPX1OPN) — Open a File” on page 181
- “read (BPX1RED) — Read from a File or Socket” on page 228
- “write (BPX1WRT) — Write to a File or Socket” on page 401.

pthread_cancel (BPX1PTB) — Cancel a Thread

BPX1PTB

thread_ID
return_value
return_code
reason_code

Purpose

Use the pthread_cancel (BPX1PTB) service to generate a cancellation request for the target thread.

Parameters

thread_ID
(input,CHAR,8) is a variable for specifying the ID of the thread to be canceled.

return_value
(output,INT,4) is a variable where the service returns `0` if the thread is canceled or the cancel is pending, or `-1` if a failure occurs.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. A successful call to pthread_cancel (BPX1PTB) generates a cancellation request for the target thread.
2. Delivery of the cancellation request caused either a nonretryable 422 abend (with reason code 01A0) or causes the signal interface routine (established with BPX1MSS) to receive control.

Example

The following code cancels the target thread. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         CALL  BPX1PTB,             pthread_cancel                   +
               (THID,               Input: Thread ID                 +
               RETVAL,              Return Value: 0, -1, or Buf length+
               RETCODE,             Return code                      +
               RSNCODE),            Reason code                      +
               VL,MF=(E,PLIST)      ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECMSERR	The service was unsuccessful due to a CMS environmental or internal error.

Return Code	Explanation
EINVAL	The *thread_ID* parameter is not valid. It does not contain a value that is consistent with thread IDs managed by the system. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRLightWeightThID.
ESRCH	The system has detected that the value specified by *thread_ID* does not refer to a thread that currently exists. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRThreadNotFound and JRAlreadyTerminated.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “pthread_create (BPX1PTC) — Create a Thread” on page 203
- “pthread_exit_and_get (BPX1PTX) — Exit and Get a New Thread” on page 209
- “pthread_join (BPX1PTJ) — Wait on a Thread” on page 212
- “pthread_kill (BPX1PTK) — Send a Signal to a Thread” on page 214
- “pthread_self (BPX1PTS) — Query Thread ID” on page 216.

pthread_create (BPX1PTC) — Create a Thread

BPX1PTC

init_rtn_addr
work_area_addr
attribute_area_addr
thread_ID
return_value
return_code
reason_code

Purpose

Use the pthread_create (BPX1PTC) service to create a new thread in the calling process. Each new thread that is created represents a single flow of control within the process with its own unique attributes.

Parameters

init_rtn_addr
(input,INT,4) is a variable for specifying the address of the initialization routine for the thread to be created. This routine is given control first when a new thread task is created to run the thread.

work_area_addr
(input,INT,4) is a variable for specifying the address of a user-supplied work area that is later passed to the initialization routine. This address is in the parameter list returned by the pthread_exit_and_get (BPX1PTX) service on a PTGETNEWTHREAD request. This parameter list is mapped by the BPXYPTXL macro. See "BPXYPTXL — Map the Parameter List for the pthread_exit_and_get Service" on page 454.

attribute_area_addr
(input,INT,4) is a variable for specifying the address of the pthread attribute area used to define the attributes of the thread to be created. If a zero address is specified, the attributes are set to their default value. For the mapping of the pthread attribute area and the definition and defaults of the supported attributes, see "BPXYPTAT — Map Attributes for the pthread_create Service" on page 453. The address of the pthread attribute area is in the parameter list returned by the pthread_exit_and_get (BPX1PTX) service on a PTGETNEWTHREAD request. This parameter list is mapped by the BPXYPTXL macro. See "BPXYPTXL — Map the Parameter List for the pthread_exit_and_get Service" on page 454.

thread_ID
(output,CHAR,8) is a variable where the service returns the thread ID for the thread that is created. This field is valid only if the service returns successfully with a return value of `0`.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if the return value is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if the return value is `-1`.

Usage Notes

Thread Initialization Routine:

1. The purpose of the thread initialization routine is to have a user-specified routine initialize the user environment for each new task that is created to process thread requests, and to control the processing of each thread that is to be run on that task.
2. The thread initialization routine is first given control when a new CMS task is created to process a thread request. At this point, the initialization routine should set up the user environment for the new task. After performing its initialization, the initialization routine can retrieve the first thread to process by invoking the pthread_exit_and_get (BPX1PTX) service.
3. This routine performs its own initialization and cleanup processing for each thread that is to be processed.
4. When this routine gains control, signals and cancellation requests are blocked.
5. Upon entry to the initialization routine, the register contents are as follows:

   R1
   Contains the address of a Type 1 parameter list. The parameter list consists of the following parameters:

   a. The address of an initial work area for use by the initialization routine during its setup processing.

   b. The address of a fullword field that contains the length of the initial work area.

   R2–R12
   Unspecified.

   R13
   Contains the address of a 144-byte save area for use by the initialization routine to allow it to perform standard save area linkage.

   R14
   Contains the return address for the initialization routine to return control to the system. This address must be preserved by the initialization routine. The high-order bit (bit 0) of this address is always ON. This bit indicates the addressing mode, which must always be AMODE(31).

   R15
   Contains the address of the initialization routine.
6. After the first thread request is received, the most efficient mechanism for the initialization routine to process subsequent thread requests is for it to call the pthread_exit_and_get (BPX1PTX) service within a loop, which causes an exit of the previous thread and the obtaining of a new thread to process.
7. To provide the most efficient interface with the high-level-language environment, the following characteristics apply to the thread initialization routine:

   a. Only one thread initialization routine is allowed per process image. When a process image is cleaned up after an invocation of the exec (BPX1EXC) service, this address can be changed. If the specified address is different within a given process image, the pthread_create (BPX1PTC) request fails with a return value of -1, a return code of EINVAL, and a reason code of JRInitRtn.

   b. The work area and pthread attribute area are passed through from pthread_create (BPX1PTC) to the caller of pthread_exit_and_get (BPX1PTX) without each being copied. Thus the caller of pthread_create (BPX1PTC) must ensure that the storage provided for these items is not released or modified prior to the use of these items by the caller of pthread_exit_and_get (BPX1PTX).

Thread IDs:

1. Threads created by pthread_create (BPX1PTC) are represented by 8-character thread IDs. A thread ID is unique only for a given process; in other words, it is possible for multiple processes to have threads represented by the same thread ID.
2. Threads to be managed by a user application should represent their threads with 8-character values, as well. To distinguish between thread IDs managed by the system and those managed by a user

application, the high-order bit of the thread ID indicates the origination of the thread ID. A thread ID managed by a user application must have its high-order bit turned on. A thread ID managed by the system has the high-order bit turned off.

3. Because thread IDs managed by the system can represent only mediumweight or heavyweight threads, those managed by a user application are considered to be lightweight threads. Any OpenExtensions service that expects a thread ID as input fails if the thread ID represents a user-application-managed, or lightweight, thread.

Other Usage Notes:

1. The pthread attribute area is passed as input to the pthread_create (BPX1PTC) service to describe the attributes of the thread to be created. The area is split into two sections. The first section is the system attribute area used by the system to build the new thread. The second section is the user area, intended for use by the thread initialization routine that receives the address of the entire pthread attribute area from pthread_exit_and_get (BPX1PTX).
2. The system offset and user offset fields indicate where the start of each area begins. The system offset field (PTATSYSOFFSET) must be set to (PTATSYSOFFVAL), or pthread_create (BPX1PTC) fails with a `-1` return value, a return code of EINVAL, and a reason code indicating the exact error. The user offset field PTATUSEROFFSET must be set to `0` if no user attributes are specified.
3. The system length and user length fields indicate the length of each area. The system length field (PTATSYSLENGTH) must be set to PTATSYSLENVAL. If not, pthread_create (BPX1PTC) fails with a `-1` return value, a return code of EINVAL, and a reason code indicating the exact error. The user length field PTATUSERLENGTH can be set to any length. However, if the sum of PTATUSERLENGTH + PTATSYSLENGTH does not equal PTATLENGTH, pthread_create (BPX1PTC) fails with a `-1` return value, a return code of EINVAL, and a reason code indicating the exact error.
4. The following describes the characteristics of each thread attribute and its impact to the pthread_create (BPX1PTC) service:
   - **Detach state** specifies the detach state of the thread to be created. A thread created in a DETACHED state cannot be joined (with the pthread_join callable service) by other threads and has its system-obtained storage freed when it exits. A thread created in an UNDETACHED state can be joined by other threads and does not have its system-obtained storage freed until it has been detached with pthread_detach. If the pthread attribute area is not specified on a pthread_create invocation, the default value is UNDETACHED.
   - **Weight** specifies the weight of the thread to be created. Both MEDIUMWEIGHT and HEAVYWEIGHT attributes result in the creation of a new CMS thread, so currently these attributes are identical.
   - **Sync type** specifies the synchronous processing type of the thread to be created. The only supported sync type is SYNCHRONOUS. A SYNCHRONOUS thread is one that is created only if the resources are immediately available to create it. An EAGAIN return code is received from a pthread_create invocation for a SYNCHRONOUS thread, if the resources are not available. If the pthread attribute area is not specified on a pthread_create invocation, the default value is SYNCHRONOUS.

Example

The following code creates a new thread. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYPTAT — Map Attributes for the pthread_create Service” on page 453.

```
         LA    R15,BUFFERA           Work area
         ST    R15,BUFA              ->above
         LA    R15,PTAT              Area mapped by BPXYPTAT
         ST    R15,PTATA             ->above
         MVC   PTATEYE,=C'BPXYPTAT'              Set the eye-catcher
         MVC   PTATLENGTH,=A(PTATUSEROFFVAL)     Length of structure
         MVC   PTATSYSOFFSET,=A(PTATSYSOFFVAL)   Sys attr offset
         MVC   PTATSYSLENGTH,=A(PTATSYSLENVAL)   Sys attr length
         MVC   PTATUSEROFFSET,=A(0)              User attr offset
         MVC   PTATUSERLENGTH,=A(0)              User attr length
         LOAD  EP=INITRTN            Get address of Init Rtn
         ST    R0,INITRTNA
         SPACE ,
```

```
         CALL  BPX1PTC,                                          +
               (INITRTNA,           Input: Init  routine address  +
               BUFA,                Input: Work area address      +
               PTATA,               Input: Attr area Address BPXYPTAT +
               THID,                Thread ID,  if Return value = 0   +
               RETVAL,              Return value: 0 or -1         +
               RETCODE,             Return code                   +
               RSNCODE),            Reason code                   +
               VL,MF=(E,PLIST)      ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EAGAIN	The system lacked the necessary resources to create the new thread.
ECMSERR	The service was unsuccessful due to a CMS environmental or internal error. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRNotAuthorized.
EFAULT	One of the parameters specified contained an address of a storage area that is not accessible to the caller.
EINVAL	One of the parameters contains a value that is not correct. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRPtatEye JRPtatSysLen JRPtatSysOff JRPtatLen JRPtatDetachState JRPtatSyncType.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “pthread_cancel (BPX1PTB) — Cancel a Thread” on page 201
- “pthread_exit_and_get (BPX1PTX) — Exit and Get a New Thread” on page 209
- “pthread_join (BPX1PTJ) — Wait on a Thread” on page 212
- “pthread_kill (BPX1PTK) — Send a Signal to a Thread” on page 214
- “pthread_self (BPX1PTS) — Query Thread ID” on page 216.

pthread_detach (BPX1PTD) — Detach a Thread

BPX1PTD

thread_ID
return_value
return_code
reason_code

Purpose

Use the pthread_detach (BPX1PTD) service to detach a thread in the calling process. When a thread is detached, its system storage can be reclaimed when the thread exits.

Parameters

thread_ID
(input,CHAR,8) is a variable for specifying the thread ID for the thread to be detached.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Example

The following code detaches a thread. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         CALL  BPX1PTD,              pthread_detach                   +
               (THID,                Input: Thread ID                 +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECMSERR	The service was unsuccessful due to a CMS environmental or internal error.
EINVAL	The value specified by thread ID is not valid, it does not contain a value that is consistent with thread IDs managed by the system. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRLightWeightThid.

Return Code	Explanation
ESRCH	The system has detected that the value specified by *thread_ID* refers to a thread that is already detached or cannot be found. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRThreadNotFound and JRAlreadyDetached.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “pthread_create (BPX1PTC) — Create a Thread” on page 203
- “pthread_join (BPX1PTJ) — Wait on a Thread” on page 212.

pthread_exit_and_get (BPX1PTX) — Exit and Get a New Thread

BPX1PTX

status_field
options_field
signal_setup_userdata
return_value
return_code
reason_code

Purpose

Use the pthread_exit_and_get (BPX1PTX) service to exit a thread, get a new thread request to process, or both. To start a new thread request, use the pthread_create (BPX1PTC) service.

Parameters

status_field
(input,INT,4) is a variable for specifying the status of the exiting thread. This status is available to any other thread that uses the pthread_join (BPX1PTJ) service to wait for the termination of this thread.

options_field
(input,INT,4) is a variable for specifying one of the following option values:

PTEXITTHREAD
Exit the calling thread. This causes the cleanup of system related resources for the calling thread.

PTGETNEWTHREAD
Exit the last obtained thread and get the next available thread to process. The first invocation of pthread_exit_and_get (BPX1PTX) from the thread initialization routine must specify this option.

PTFAILIFLASTTHREAD
Exit the calling thread only if it is not the last thread in the process.

The default option value is PTEXITTHREAD. The option values are defined in the BPXYCONS macro. See "BPXYCONS — Map Constants" on page 417. These options can be combined by specifying them with a plus between them.

signal_setup_userdata
(input,CHAR,4) is a variable for specifying 4 bytes of user data normally supplied on the signal setup service, cmssigsetup (BPX1MSS). This field is used only when the PTGETNEWTHREAD option is specified. If this field contains a zero address, the signal setup user data is not changed for this thread. This field is ignored when the PTEXITTHREAD option is specified.

return_value
(output,INT,4) is a variable where the service stores the return value, which varies depending on the options specified, as follows:

- PTEXITTHREAD option specified:

  -1
  The caller asked to exit the calling thread, but the thread could not be exited. For an explanation of the error, see the return code and reason code.

  0
  The thread was successfully exited.

- PTGETNEWTHREAD option specified:

-1
The caller asked for a new thread to process, but the thread request could not be satisfied. No new thread requests can be handled by the calling task. For an explanation of the error, see the return code and reason code.

>0
The address of the parameter list for the new thread request to be processed. The parameter list consists of the following:

- The user work area address specified on the pthread_create (BPX1PTC) call.
- The user attribute area address specified on the pthread_create (BPX1PTC) call.
- The address of an 8-byte field that contains the thread ID of the thread request.
- The address of a 4-byte thread run status field.

This parameter list is mapped by the BPXYPTXL macro. See "BPXYPTXL — Map the Parameter List for the pthread_exit_and_get Service" on page 454. The storage for the list is supplied by the system and should not be modified or freed by the caller of pthread_exit_and_get (BPX1PTX).

- PTFAILIFLASTTHREAD option specified:

-1
The caller asked to edit the calling thread only if it was not the last thread, but the thread could not be exited. See the return code and reason code for an explication of the error.

0
The thread was successfully exited.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. The first call to pthread_exit_and_get (BPX1PTX) from the thread initialization routine must specify the PTGETNEWTHREAD option. On this first call, a thread request is retrieved without causing a thread exit to occur. All subsequent calls to the service result in a thread exit and then obtaining the next available thread request. If the PTGETNEWTHREAD option is not specified on the first call, the service fails with a -1 return value, an EINVAL return code, and a JRGetFirst reason code.
2. Using the PTGETNEWTHREAD option can cause failure if the process is being quiesced. If this happens, the pthread_exit_and_get (BPX1PTX) service fails with a -1 return value, an EINVAL return code, and a JRQuiesceInProgress reason code.. At this point, the caller should perform its own cleanup and return to the operating system to allow the task to terminate.
3. If the PTFAILIFLASTTHREAD option is specified, and pthread_exit_and_get (BPX1PTX) is issued from the last thread, the thread is not exited. The service fails with a -1 return value, an EINVAL return code, and a JrLastThread reason code. Any thread that has never issued a pthread_create or was not created with pthread_create is considered the last thread when using the PTFAILIFLASTTHREAD option.
4. When pthread_exit_and_get (BPX1PTX) is used to get a new thread request, the signal environment is inherited from the creator of the thread. The signal state for the newly created thread is roughly analogous to that of a newly created process after the spawn service has been performed. The one exception is that the new thread inherits the setup state from the creator.
5. A successful call to pthread_exit_and_get (BPX1PTX) awakens a thread that has used the pthread_join (BPX1PTJ) service to wait for the exiting thread. The thread exit status specified on the pthread_exit_and_get (BPX1PTX) call is made available to the waiting thread.

6. If pthread_exit_and_get (BPX1PTX) fails for any reason (with a return value of -1), the caller should perform cleanup and return to the operating system to allow the task to end.
7. When this service is called from the initial pthread, it waits for all threads created with pthread_create to end.
8. For information about the pthread attribute area, see "pthread_create (BPX1PTC) — Create a Thread" on page 203.

Example

The following code exits a thread. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551.

```
         CALL  BPX1PTX,              pthread_exit_and_get             +
               (STATFLD,             Input: Status field              +
               OPTIONS,              Input: Options field             +
               SIGNALREG,            Input: Signal setup usrdata+
               RETVAL,               Return value: 0 or -1  ->BPXYPTXL +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECMSERR	The service was unsuccessful due to a CMS environmental or internal error. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRAlreadyExited.
EINVAL	One of the parameters contains a value that is not valid. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRInvOption, JRGetFirst, and JRHeavyWeight.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, "Return Codes," on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, "Reason Codes," on page 495.

Related Services

Other callable services related to this service are:

- "pthread_create (BPX1PTC) — Create a Thread" on page 203
- "pthread_join (BPX1PTJ) — Wait on a Thread" on page 212.

pthread_join (BPX1PTJ) — Wait on a Thread

BPX1PTJ

thread_ID
status_field_address
return_value
return_code
reason_code

Purpose

Use the pthread_join (BPX1PTJ) service to obtain the termination status for a specific thread. This service waits only if the thread has not ended, is not in a detached state, and is not currently joined by another thread.

Parameters

thread_ID
(input,CHAR,8) is a variable for specifying the thread ID for the thread to be waited upon.

status_field_addr
(input,INT,4) is a variable for specifying the address of a status field where the service returns the exit status of the thread specified by the *thread_ID* parameter. If this field is zero, the thread exit status is not returned. This field is mapped by the BPXYWAST macro. See "BPXYWAST — Map the Wait Status Word" on page 486.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Note

The pthread_join (BPX1PTJ) service can be called repeatedly for a thread until it is detached. However, a thread can be the target of only one pthread_join call at a time.

Example

The following code waits on a thread. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551.

```
         CALL  BPX1PTJ,              pthread_join                     +
               (THID,                Input: Thread ID                 +
               =A(0),                Input: ->Status Field or 0       +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECMSERR	The service was unsuccessful due to a CMS environmental or internal error.
EDEADLK	A deadlock was detected, or the value specified by *thread_ID* refers to the calling thread. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRJoinLoop and JRJoinToSelf.
EFAULT	One of the parameters specified contained an address of a storage area that is not accessible to the caller. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRJoinExitStatPtr.
EINTR	The calling process received a signal prior to the completion of an event that would cause the pthread_join (BPX1PTJ) service to return. The service was interrupted by a signal. In this case, the value contained in *status_field_address* is undefined.
EINVAL	The value specified by thread ID is not valid; it does not contain a value that is consistent with thread IDs managed by the system. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRLightWeightThread.
ESRCH	The value specified by *thread_ID* does not refer to a thread that is undetached. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRThreadNotFound, JRAlreadyJoined, and JRAlreadyDetached.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “pthread_create (BPX1PTC) — Create a Thread” on page 203
- “pthread_detach (BPX1PTD) — Detach a Thread” on page 207.

pthread_kill (BPX1PTK) — Send a Signal to a Thread

BPX1PTK

thread_ID
signal
signal_options
return_value
return_code
reason_code

Purpose

Use the pthread_kill (BPX1PTK) service to target a signal to a particular thread. This service is limited to interthread communication within a process.

Parameters

thread_ID
(input,CHAR,8) is a variable for specifying the thread ID for the thread to receive the signal.

signal
(input,INT,4) is a variable for specifying the signal number to be sent to the thread indicated by the *thread_ID* parameter. This must be one of the signals defined in the BPXYSIGH macro, or `0`.

If the signal is `0`, error checking takes place but no signal is sent. You can call the pthread_kill (BPX1PTK) service with a signal value of `0` to verify the *thread_ID* parameter is correct before you actually send the signal.

signal_options
(input,BINARY,4) is a variable for specifying the binary flags that describe how the signal is to be handled by both the OpenExtensions kernel and the user-supplied signal interface routine (SIR). The signaling options are passed to the SIR in the signal information control block mapped by the BPXYPPSD macro. See "BPXYPPSD — Map the Signal Delivery Data Structure" on page 451. The *signal_options* parameter is mapped as follows:

First 2 bytes
User-defined bytes delivered with the signal to the SIR in the signal information control block. These bytes are mapped by the BPXYPPSD macro.

Last 2 bytes
Reserved

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Example

The following code signals a thread. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
*        MVC   SIGNAL,=A(SIGALRM#)   Input: SIGALRM           BPXYSIGH
*        MVC   SIGNALOPTIONS,=XL4'00000000' Input: Signal options
*        CALL  BPX1PTK,              pthread_kill                      +
               (THID,                Input: Thread ID                  +
               SIGNAL,               Input: Signal or 0       BPXYSIGH +
               SIGNALOPTIONS,        Input: Signal options             +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

VM-Related Information

See Appendix E, “The Relationship of OpenExtensions Signals to Callable Services,” on page 557.

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECMSERR	The service was unsuccessful due to a CMS environmental or internal error. Consult the reason code to determine the exact reason the error occurred.
EINVAL	One of the following conditions causes this return code: • The value of *signal* is not valid or is not the number of a supported signal. • The thread corresponding to *thread_ID* was not found, is not valid, or has ended. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRInvalidSignal, JRLightWeightThid, JRThreadNotFound, and JRThreadTerm.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “kill (BPX1KIL) — Send a Signal to a Process” on page 146
- “sigaction (BPX1SIA) — Examine or Change a Signal Action” on page 315.

pthread_self (BPX1PTS) — Query Thread ID

BPX1PTS

thread_ID

Purpose

Use the pthread_self (BPX1PTS) service to get the thread ID of the calling thread.

Parameters

thread_ID
(output,CHAR,8) is a variable where the service returns the thread ID of the calling thread.

Usage Notes

1. The caller should invoke this service only once when needing the thread ID of the active thread. It should save a copy of the thread ID in its own storage for repetitive usage.
2. If this service fails, the calling thread abends.

Example

The following code gets the thread ID of the calling thread. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551.

```
         CALL  BPX1PTS,              pthread_self                    +
               (THID),               Output: Thread ID               +
               VL,MF=(E,PLIST)       ---------------------------------
```

Related Service

Another callable service related to this service is:

- "pthread_create (BPX1PTC) — Create a Thread" on page 203.

pthread_setintr (BPX1PSI) — Examine and Change Interrupt State

BPX1PSI

interrupt_state
return_value
return_code
reason_code

Purpose

Use the pthread_setintr (BPX1PSI) service to set the interruptibility state of the calling thread and atomically return the previous interruptibility state.

Parameters

interrupt_state
(input,INT,4) is a variable for specifying the interrupt state to be set. The following constants defined in the BPXYCONS macro define the valid states. See "BPXYCONS — Map Constants" on page 417.

Constant
Description

PTHREAD_INTR_ENABLE#
Enables interruptibility, so new or pending cancellation requests against the target thread are acted upon according to the interruptibility type set by the pthread_setintrtype (BPX1PST) service.

PTHREAD_INTR_DISABLE#
Disables interruptibility, so cancellation requests against the target thread are held pending.

return_value
(output,INT,4) is a variable where the service returns the previous interrupt state, or -1 if the service did not complete successfully.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. Setting the interruptibility state allows a user to control when cancellation requests sent by the BPX1PTB service are handled.
2. BPX1PSI and BPX1PST establish three interruptibility states:

   Disabled
   Cancellation requests are left pending.

   Controlled
   Cancellation requests are left pending until the next cancellation point is reached. Cancellation points are defined as:

   - When the pthread_testintr (BPX1PTI) service is invoked.
   - When a thread is placed in an unbounded wait during a call of an OpenExtensions service. Some examples of these types of calls are

  - "cond_timed_wait (BPX1CTW) — Suspend a Thread for a Limited Time or for an Event" on page 52
  - "pause (BPX1PAS) — Suspend a Process Pending a Signal" on page 197
  - "sleep (BPX1SLP) — Suspend Execution of a Process for an Interval of Time" on page 328
  - "sigsuspend (BPX1SSU) — Change the Signal Mask and Suspend the Thread Until a Signal Is Delivered" on page 324
  - "sigwait (BPX1SWT) — Wait for a Signal" on page 326.

  Asynchronous
  Cancellation requests can be delivered any time.

3. The default interrupt state for newly created threads and the initial thread is PTHREAD_INTR_ENABLE#.
4. The default interrupt type for newly created threads and the initial thread is PTHREAD_INTR_CONTROLLED#.
5. The interrupt types of controlled and asynchronous are set with the pthread_setintrtype (BPX1PST) service. See "pthread_setintrtype (BPX1PST) — Examine and Change Interrupt Type" on page 219. These states are acted upon only if thread interruption is enabled. If a cancellation request is pending and the interrupt state or type is set to allow asynchronous cancellation requests, the thread is canceled before control is returned to the invoker.

Example

The following code examines and changes the interrupt state of the calling thread. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551.

```
         CALL  BPX1PSI,              Examine and change interrupt state+
               (INTRSTATE,           Input: Interrupt state   BPXYCONS +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return code:

Return Code	Explanation
EINVAL	One of the parameters contains a value that is not valid.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, "Return Codes," on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, "Reason Codes," on page 495.

Related Services

Other callable services related to this service are:

- "pthread_cancel (BPX1PTB) — Cancel a Thread" on page 201
- "pthread_setintrtype (BPX1PST) — Examine and Change Interrupt Type" on page 219
- "pthread_testintr (BPX1PTI) — Cause a Cancellation Point to Occur" on page 221.

pthread_setintrtype (BPX1PST) — Examine and Change Interrupt Type

BPX1PST

interrupt_type
return_value
return_code
reason_code

Purpose

Use the pthread_setintrtype (BPX1PST) service to set the interruptibility type of the calling thread and atomically return the previous interruptibility type.

Parameters

interrupt_type
(input,INT,4) is a variable for specifying the interrupt type to be set. The following constants defined in the BPXYCONS macro define the valid states.

Constant
Description

PTHREAD_INTR_ASYNCHRONOUS#
When interruptibility is enabled, cancellation requests can be acted upon any time.

PTHREAD_INTR_CONTROLLED#
When interruptibility is enabled, cancellation requests are held pending until a cancellation point is reached. See the usage notes for the definition of cancellation points.

return_value
(output,INT,4) is a variable where the service returns the previous interrupt type, or -1 if the service did not complete.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. The default interrupt type for newly created threads and the initial thread is PTHREAD_INTR_CONTROLLED#. If a cancellation request is pending and the interrupt state is set to PTHREAD_INTR_ASYNCHRONOUS#, the cancellation request is acted upon before control is returned to the invoker.
2. For more information on controlling cancellation requests, see the usage notes for “pthread_setintr (BPX1PSI) — Examine and Change Interrupt State” on page 217.

Example

The following code examines and changes the interrupt type of the calling thread. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         CALL  BPX1PST,              Examine and change interrupt type +
               (INTRTYPE,            Input: Interrupt type    BPXYCONS +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return code:

Return Code	Explanation
EINVAL	One of the parameters contains a value that is not valid.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “pthread_cancel (BPX1PTB) — Cancel a Thread” on page 201
- “pthread_setintr (BPX1PSI) — Examine and Change Interrupt State” on page 217
- “pthread_testintr (BPX1PTI) — Cause a Cancellation Point to Occur” on page 221.

pthread_testintr (BPX1PTI) — Cause a Cancellation Point to Occur

BPX1PTI

return_value

return_code

reason_code

Purpose

Use the pthread_testintr (BPX1PTI) service to cause a cancellation point to occur. If a cancellation request is pending, the pending request is acted upon before this service returns.

Parameters

return_value
(output,INT,4) is a variable where the service returns a `0` if the thread did not have any pending cancellation requests, or `-1` if the service did not complete (the cancellation request was not tested).

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. If a cancellation request is pending at the time of the invocation of this service, control is not returned.
2. Invoking the pthread_testintr service does not affect the interrupt state or type.
3. For more information on using this service, see the usage notes for "pthread_setintr (BPX1PSI) — Examine and Change Interrupt State" on page 217.

Example

The following code causes a cancellation point to occur. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551.

```
         CALL  BPX1PTI,              Cause an interrupt point to occur +
               (RETVAL,              Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

For a complete list of return codes for OpenExtensions callable services, see Appendix A, "Return Codes," on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, "Reason Codes," on page 495.

Related Services

Other callable services related to this service are:

- "pthread_cancel (BPX1PTB) — Cancel a Thread" on page 201
- "pthread_setintr (BPX1PSI) — Examine and Change Interrupt State" on page 217

- “pthread_setintrtype (BPX1PST) — Examine and Change Interrupt Type” on page 219.

queue_interrupt (BPX1SPB) — Return the Last Interrupt Delivered

BPX1SPB

return_value
return_code
reason_code

Purpose

Use the queue_interrupt (BPX1SPB) service to return the last interrupt delivered to the signal interface routine (SIR) back to the OpenExtensions kernel. The interrupt can be a signal, a cancellation request, or a quiesce request.

Parameters

return_value
(output,INT,4) is a variable where the function returns 0 if it has permission to return the specified interrupt for delivery at the next kernel call. If no interrupt is returned, -1 is returned.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. The the data mapped by the PPSD will be used by the queue_interrupt service and, therefore, should not be modified by the invoker as this may result in an EINVAL.
2. The queue_interrupt service will return the interrupt back to the OpenExtensions kernel and restore the signal blocking mask to its pre-interrupt state. The interrupt will then be delivered to this thread upon the next syscall invocation.

Characteristics and Restrictions

The intended use of the queue_interrupt (BPX1SPB) service is from the signal interface routine specified on "cmssigsetup (BPX1MSS) — Set Up CMS Signals" on page 40. Although the queue_interrupt service can be used anywhere, all signals must be blocked and the task must have setup signals by invoking the cmssigsetup service before calling queue_interrupt. See Appendix E, "The Relationship of OpenExtensions Signals to Callable Services," on page 557.

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EINVAL	The value of Signal in the PPSD at the time this service was invoked was an unsupported signal. Either there was a storage overlay in the PPSD, or no signal was ever delivered to this task.

Return Code	Explanation
EPERM	The caller does not have permission to return the interrupt now. All signals must be blocked, and the task must invoke cmssigsetup (BPX1MSS) before the queue_interrupt service is invoked. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRSignalsNotBlocked and JRNotSigsetup.

The following code uses the queue_interrupt to return the last signal delivered to the signal interface routine (SIR). This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         CALL  BPX1SPB,              Queue the signal                +
               (RETVAL,              Return value: 0 or -1           +
               RETCODE,              Return code                     +
               RSNCODE),             Reason code                     +
               VL,MF=(E,PLIST)       ---------------------------------
```

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “cmssigsetup (BPX1MSS) — Set Up CMS Signals” on page 40
- “pthread_cancel (BPX1PTB) — Cancel a Thread” on page 201.

quiesce_threads (BPX1PTQ) — Quiesce Threads in a Process

BPX1PTQ

quiesce_type
user_data
return_value
return_code
reason_code

Purpose

Use the quiesce_threads (BPX1PTQ) service to perform one of the following functions:

- Synchronously quiesce the initial thread of the process and all threads created with the pthread_create (BPX1PTC) service
- Query the thread environment in the current process

Parameters

quiesce_type
(input,INT,4) is a variable for specifying one of the following values to indicate the type of function to be performed:

QUIESCE_TERM
Quiesce the initial thread and all threads created with pthread_create, allowing the signal interface routine to receive control when the quiesce request is delivered.

QUIESCE_FORCE
Quiesce the initial thread and all threads created with pthread_create, not allowing the signal interface routine to receive control when the quiesce request is delivered.

QUIESCE_QUERY
Count the number of POSIX threads in the current process, which includes the initial thread of the process and all threads created with pthread_create, and return the count in *return_value*.

The quiesce_type values are defined in the BPXYCONS macro. See "BPXYCONS — Map Constants" on page 417.

user_data
(input,CHAR,4) is a variable for specifying user data to be passed to the signal interface routine when the quiesce request is delivered.

return_value
(output,INT,4) is a variable where the service returns a value that depends on the *quiesce_type* specified:

- For QUIESCE_TERM or QUIESCE_FORCE:

  -1
  The caller asked to quiesce all threads in the current process, but all threads may not have been quiesced. For an explanation of the error, see the return code and reason code.

  0
  All threads in the current process are successfully quiesced.

- For QUIESCE_QUERY:

  -1
  The caller asked to query the number of threads in the process, but the request could not be completed. For an explanation of the error, see the return code and reason code.

1
The calling thread is the initial thread, and no threads created with pthread_create exist in the current process.

>1
This is the count of all the POSIX threads in the current process (the initial thread plus all threads created with pthread_create).

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. Invoking quiesce_threads (BPX1PTQ) delivers a quiesce request to the initial thread and all threads created with pthread_create in the process. When *quiesce_type* is QUIESCE_TERM, the request is delivered to each thread by the signal interface routine (SIR), if the process is set up to intercept the quiesce request. If the process is not set up for quiesce request interception, or if *quiesce_type* is QUIESCE_FORCE, the CMS OpenExtensions kernel performs the quiesce request for each thread. For details on how to intercept quiesce requests, see "cmssigsetup (BPX1MSS) — Set Up CMS Signals" on page 40.
2. When quiescing threads before an exec (BPX1EXC) call, the quiesce_threads service should be invoked from the exec user exit. This invocation allows the probable success of the exec to be determined before all other threads in the process are quiesced.
3. The quiesce_threads service should be invoked before an _exit (BPX1EXI) call to prevent the other threads in the process from receiving an asynchronous abend. The quiesce_threads service ends the other threads in the CMS OpenExtensions kernel, preventing them from being asynchronously abended at an unknown point.
4. The quiesce_threads service posts all threads that are in pthread_exit_and_get (BPX1PTX) waiting for more work. The pthread_exit_and_get service returns to the invoker with a -1 return value. The invoker can then clean up the related resources before the normal end of the thread.

Example

The following code terminates all other pthreads in the caller's process. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structure, see "BPXYCONS — Map Constants" on page 417.

```
         CALL  BPX1PTQ,              pthread_quiesce                   +
               (=A(QUIESCE_TERM),    Input: Quiesce type      BPXYCONS +
               =A(0),                Input: User data - Catch data PPSD+
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return one of the following return codes:

Return Code	Explanation
ECMSERR	A CMS environment or internal error has occurred. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRQuiesceInProgress.

Return Code	Explanation
EINTR	The quiesce was interrupted by a signal before all threads were quiesced.
EINVAL	The value specified for *quiesce_type* was incorrect. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRQuiesceTypeInvalid.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “pthread_create (BPX1PTC) — Create a Thread” on page 203
- “cmssigsetup (BPX1MSS) — Set Up CMS Signals” on page 40.

read (BPX1RED) — Read from a File or Socket

BPX1RED

file_descriptor
buffer_address
buffer_ALET
read_count
return_value
return_code
reason_code

Purpose

Use the read (BPX1RED) service to read a specified number of bytes from a file or socket into a buffer that you provide.

Note: The read service is not related to the `read` shell command.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the file descriptor of the file or socket to be read. The file must be open.

buffer_address
(input,INT,4) is a variable for specifying the address of the buffer into which data is to be read.

buffer_ALET
(input,INT,4) is a variable for specifying the access list entry token (ALET) for the buffer.

Note: This parameter is ignored.

read_count
(input,INT,4) is a variable for specifying the number of bytes you want to read from the file or socket. This number must be less than or equal to the length of the buffer you provide for data to be read into.

return_value
(output,INT,4) is a variable where the service returns the number of bytes actually read (may be 0) if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

Access Time — A successful read updates the access time of the file read.

Origin of Bytes Read — If the file specified by *file_descriptor* is a regular file, or any other type of file where a seek operation is possible, bytes are read from the file offset associated with the file descriptor. A successful read increments the file offset by the number of bytes read.

For files where no seek operation is possible, there is no file offset associated with the file descriptor. Reading begins at the current position in the file.

Number of Bytes Read — The value of *read_count* is not checked against any system limit, although a limit can be imposed by a high-level-language POSIX implementation.

When a read request completes, the *return_value* field shows the number of bytes actually read—a number less than or equal to the number specified as *read_count*. The following are some reasons why the number of bytes read might be less than the number of bytes requested:

- Fewer than the requested number of bytes remained in the file; the end of file was reached before *read_count* bytes were read.
- The service was interrupted by a signal after some but not all of the requested bytes were read. (If no bytes were read, the return value is set to -1 and an error is reported.)
- The file is a pipe, FIFO, or special file and fewer bytes than *read_count* specified were available for reading.

There are several reasons why a read request may complete successfully with no bytes read — that is, with *return_value* set to 0. For example, zero bytes are read in these cases:

- The service specified a *read_count* of zero.
- The starting position for the read was at or beyond the end of the file.
- The file being read is a FIFO file or a pipe, and no process has the pipe open for writing.
- The file being read is a terminal and a zero-length canonical file was read.

Nonblocking — If a process has a pipe open for writing with nonblocking specified, a request to read from the file ends with a return value of -1 and a "Resource temporarily unavailable" return code. But if nonblocking was not specified, the read request is blocked (does not return) until some data is written or the pipe is closed by all other processes that have the pipe open for writing.

Terminals operate this way too, except how they act depends on how they were opened. If the terminal is opened blocking, the reads are blocked if there is no data. If it is opened nonblocking, EAGAIN is returned if there is no data.

SIGTTIN Processing — This service causes signal **SIGTTIN** to be sent if all the following conditions are met:

- The process is attempting to read from its controlling terminal.
- The process is running in a background process group.
- The **SIGTTIN** signal is not blocked or ignored.
- The process group of the process is not orphaned.

If **SIGTTIN** has a handler, the handler gets control and the read ends with a return code of EINTR. If **SIGTTIN** is set to default, the process stops in the read and continues when the process is moved to the foreground.

Characteristics and Restrictions

If the file was opened by an authorized program, all subsequent reads and writes against the file must be issued from an authorized state.

Example

The following code reads 80 bytes from the specified file (FILEDESC) and place them in the area provided (BUFFERA). This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   FILEDESC,..          File descriptor
         LA    R15,BUFFERA          Buffer
         ST    R15,BUFA             Buffer address
         MVC   BUFLENA,=F'80'       Read buffer length
         SPACE ,
         CALL  BPX1RED,             Read from a file                +
               (FILEDESC,           Input: File descriptor          +
               BUFA,                ->Buffer to read into           +
```

```
                PRIMARYALET,          Input: Buffer ALET                +
                BUFLENA,              Input: Number of bytes to read    +
                RETVAL,               Return value: 0, -1, or char count+
                RETCODE,              Return code                       +
                RSNCODE),             Reason code                       +
                VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EAGAIN	The file was opened with the nonblock option and data is not available to be read.
EBADF	The *file_descriptor* parameter does not contain the descriptor of an open file, or the file is not opened for read. The following reasons codes can accompany this return code: JRFileDesNotInUse, JRFileNotOpen.
EINTR	The service was interrupted by a signal before it could read any data.
EINVAL	The *read_count* parameter contains a value that is less than zero. The following reason code can accompany this return code: JRSocketCallParmError.
EIO	The process is in a background process group and is attempting to read from its controlling terminal. Either the process is ignoring or blocking the **SIGTTIN** signal, or the process group is orphaned.
ENOBUFS	A buffer could not be obtained.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “fcntl (BPX1FCT) — Control Open File Descriptors” on page 88
- “lseek (BPX1LSK) — Change the File Offset” on page 154
- “open (BPX1OPN) — Open a File” on page 181
- “pipe (BPX1PIP) — Create an Unnamed Pipe” on page 199
- “write (BPX1WRT) — Write to a File or Socket” on page 401

readdir (BPX1RDD) — Read an Entry from a Directory

BPX1RDD

directory_file_descriptor
buffer_address
buffer_ALET
buffer_length
return_value
return_code
reason_code

Purpose

Use the readdir (BPX1RDD) service to read multiple name entries from a directory.

Parameters

directory_file_descriptor
(input,INT,4) is a variable for specifying the directory file descriptor for the directory from which entries are to be read. This value was returned by the opendir (BPX1OPD) callable service when the directory was opened.

buffer_address
(input,INT,4) is a variable for specifying the address of the buffer in which the service is to write the directory entries. The directory entries are mapped by the BPXYDIRE macro. See "BPXYDIRE — Map Directory Entries for the readdir Service" on page 420.

buffer_ALET
(input,INT,4) is a variable for specifying the access list entry token (ALET) for *buffer_address* that identifies the address space or data space where the buffer resides.

Note: This parameter is ignored in the OpenExtensions implementation.

buffer_length
(input,INT,4) is a variable for specifying the length in bytes of the buffer pointed to by *buffer_address*.

return_value
(output,INT,4) is a variable where the service returns the number of directory entries read into the buffer if the service is successful, or `-1` if unsuccessful. A value of `0` indicates the end of the directory.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. This interface differs from the POSIX C high-level-language interface in that it returns more than one directory entry, and it also returns the entries in the caller's buffer.
2. The buffer contains a variable number of variable-length directory entries. Only full entries are placed in the buffer, up to the buffer size specified, and the number of entries is returned.
3. Each directory entry returned has the following format:

entry_length
A 2-byte field that specifies the total length of the entry, including this field.

name_length
A 2-byte field that specifies the length of the following *member_name* field.

member_name
A character field of length *name_length*. This name is not null-terminated.

file_system_specific_data
If *name_length* + 4 = *entry_length*, this field is not present.

The entries are packed together, and the length fields are not aligned on any particular boundary.

4. The buffer returned by one call to the readdir (BPX1RDD) service must be used again on the next call to the readdir service to continue reading entries from where you left off. The buffer must not be altered between calls, unless the directory has been rewound.
5. The end of the directory is indicated in either of two ways:
   - A *return_value* of 0 entries is returned.
   - Some physical file systems may return a null name entry as the last entry in the caller's buffer. A null name entry has an *entry_length* of 4 and a *name_length* of 0.

   Both conditions should be checked for by the caller of the readdir (BPX1RDD) service.

Example

The following code reads multiple name entries from the specified directory (DIRECTDES). This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYDIRE — Map Directory Entries for the readdir Service” on page 420.

```
         MVC   DIRECTDES,..          Directory descriptor from  opendir
         LA    R15,BUFFERA
         ST    R15,BUFA
         MVC   BUFLENA,=F'1023'
         CALL  BPX1RDD,              Read entries from a directory     +
               (DIRECTDES,           Input: Directory file descriptor  +
               BUFA,                 Output: ->buffer         BPXYDIRE +
               PRIMARYALET,          Input: buffer ALET                +
               BUFLENA,              Input: buffer size                +
               RETVAL,               Return value: 0, -1, entries read +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	The *directory_file_descriptor* argument does not refer to an open directory.
EINVAL	The buffer is too small to contain any entries.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “closedir (BPX1CLD) — Close a Directory” on page 36
- “opendir (BPX1OPD) — Open a Directory” on page 185

- “rewinddir (BPX1RWD) — Reposition a Directory Stream to the Beginning” on page 254.

read_external_link (BPX1RXL) — Read the Contents of a CMS External Link

BPX1RXL

link_name_length
link_name
buffer_length
buffer_address
return_value
return_code
reason_code

Purpose

Use the read_external_link (BPX1RXL) service to read the contents of a CMS external link into a buffer that you provide. The external link contains the data that was specified when the external link was defined by the create_external_link (BPX1ELN) service.

Parameters

link_name_length
(input,INT,4) is a variable for specifying the length of the *link_name* parameter.

link_name
(input,CHAR,*link_name_length*) is a variable for specifying the name of the external link to be read.

buffer_length
(input,INT,4) is a variable for specifying the length in bytes of the buffer pointed to by the *buffer_address* parameter.

buffer_address
(input,INT,4) is a variable for specifying the address of the buffer where the service is to return the contents of the external link.

return_value
(output,INT,4) is a variable where the service returns a count of the number of characters placed in the buffer if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. If the buffer supplied to BPX1RXL is too small to hold the contents of the external link, the contents are truncated to the length of the buffer (*buffer_length*). If the value returned in *return_value* is the length of the buffer, you can use the lstat (BPX1LST) service to determine the actual length of the external link. See "lstat (BPX1LST) — Get Status Information about a File or Symbolic Link by Path Name" on page 157.
2. If *buffer_length* is specified as 0, the value returned in *return_value* is the number of bytes in the external link, and the buffer remains unchanged.

Example

The following code reads the contents of an external link named /u/dpt37/payroll into the buffer provided. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   BUFFERB(19),=CL16'/u/dpt37/payroll'
         MVC   BUFLENB,=F'16'
         LA    R15,BUFFERA
         ST    R15,BUFA
         MVC   BUFLENA,=F'1023'
         SPACE ,
         CALL  BPX1RXL,              Read contents of an external link +
               (BUFLENB,             Input: Linkname length            +
               BUFFERB,              Input: Link name                  +
               BUFLENA,              Input: Buffer size - 1023         +
               BUFA,                 ->Buffer for external link        +
               RETVAL,               Return value: 0, -1 or char count +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	Search permission is denied for a component of the specified external link.
EINVAL	The file identified by *link_name* is not an external link, or there was a problem with the supplied buffer. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRFileNotExtLink and JRRd1BuffLenInvalid.
ELOOP	A loop exists in the Mount External Links (MELs) encountered during resolution of the *link_name* argument, if more than eight MELs are detected.
ENAMETOOLONG	The *link_name* parameter is longer than 1023 characters, or some component of the link name is longer than 255 characters. CMS does not support name truncation.
ENOENT	No file named *link_name* was found. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRFileNotThere.
ENOTDIR	A component of the path prefix is not a directory.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service is:

- “lstat (BPX1LST) — Get Status Information about a File or Symbolic Link by Path Name” on page 157
- “symlink (BPX1SYM) — Create a Symbolic Link to a Path Name” on page 345
- “unlink (BPX1UNL) — Remove a Directory Entry” on page 379.

readlink (BPX1RDL) — Read the Value of a Symbolic Link

BPX1RDL

link_name_length
link_name
buffer_length
buffer_address
return_value
return_code
reason_code

Purpose

Use the readlink (BPX1RDL) service to read the contents of a symbolic link into a buffer that you provide. The symbolic link contains the path name that was specified when the symbolic link was defined by the symlink (BPX1SYM) service.

Parameters

link_name_length
(input,INT,4) is a variable for specifying the length of the *link_name* parameter.

link_name
(input,CHAR,*link_name_length*) is a variable for specifying the link name of the symbolic link to be read.

buffer_length
(input,INT,4) is a variable for specifying the length in bytes of the buffer pointed to by the *buffer_address* parameter.

buffer_address
(input,INT,4) is a variable for specifying the address of the buffer where the service is to return the value of the symbolic link. The value of the symbolic link is actually the path name that was specified when the symbolic link was created. The buffer must reside in the process's address space.

return_value
(output,INT,4) is a variable where the service returns a count of the number of characters placed in the buffer if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. If the buffer supplied to readlink (BPX1RDL) is too small to contain the value of the symbolic link, the value is truncated to the length of the buffer (*buffer_length*). If the value returned is the length of the buffer, you can use the lstat (BPX1LST) service to determine the actual length of the symbolic link. See "lstat (BPX1LST) — Get Status Information about a File or Symbolic Link by Path Name" on page 157.
2. If the *buffer_length* is 0, the value returned is the number of bytes in the symbolic link and the buffer remains unchanged.

Example

The following code reads the contents of symbolic link **/personnel/templink** into the buffer provided. This will be the path name that was specified when the symbolic link was defined. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551.

```
         MVC   BUFFERB(19),=CL19'/personnel/templink'
         MVC   BUFLENB,=F'19'
         LA    R15,BUFFERA
         ST    R15,BUFA
         MVC   BUFLENA,=F'1023'
         SPACE ,
         CALL  BPX1RDL,              Read the value of a symbolic link +
               (BUFLENB,             Input: Linkname length            +
               BUFFERB,              Input: Link name                  +
               BUFLENA,              Input: Buffer size - 1023         +
               BUFA,                 ->Buffer for symbolic link        +
               RETVAL,               Return value: 0, -1 or char count +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCESS	Search permission is denied for a component of the path prefix.
EINVAL	The file named by *link_name* is not a symbolic link or there was a problem with the supplied buffer. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRFileNotSymLink, and JRRdlBuffLenInvalid.
ELOOP	A loop exists in symbolic links encountered during resolution of the *link_name* argument. This error is issued if more than 8 symbolic links are detected in the resolution of *link_name*.
ENAMETOOLONG	The *link_name* parameter is longer than 1023 characters, or some component of the link name is longer than 255 characters. CMS does not support name truncation.
ENOENT	No file with the name specified by *link_name* was found. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRFileNotThere.
ENOTDIR	A component of the path prefix is not a directory.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, "Return Codes," on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, "Reason Codes," on page 495.

Related Services

Other callable services related to this service are:

- "lstat (BPX1LST) — Get Status Information about a File or Symbolic Link by Path Name" on page 157
- "symlink (BPX1SYM) — Create a Symbolic Link to a Path Name" on page 345
- "unlink (BPX1UNL) — Remove a Directory Entry" on page 379.

readv (BPX1RDV) — Read Data and Store It in a Set of Buffers

BPX1RDV

socket_descriptor
IOV_count
IOV_structures
IOV_ALET
IOV_buffer_ALET
return_value
return_code
reason_code

Purpose

Use the readv (BPX1RDV) service to read data from a socket and store it in a set of buffers.

Parameters

socket_descriptor
(input,INT,4) is a variable for specifying the descriptor of the socket.

IOV_count
(input,INT,4) is a variable for specifying the number of buffers that are pointed to by *IOV_structure*.

IOV_structures
(input,CHAR,*IOV_count* times length of BPXYIOV) is a variable for specifying the IOV structures that contain information about the buffers in which data is to be stored. The IOV structure is mapped by the BPXYIOV macro. See "BPXYIOV — Map the I/O Vector Structure" on page 430.

IOV_ALET
(input,INT,4) is a variable for specifying the access list entry token (ALET) for *IOV_structures*.

Note: This parameter is ignored.

IOV_buffer_ALET
(input,INT,4) is a variable for specifying the ALET for the buffers that are pointed to by *IOV_structures*.

Note: This parameter is ignored.

return_value
(output,INT,4) is a variable where the service returns the number of bytes that were read into the buffers if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

Socket Files — When used for datagram sockets, this service returns the entire datagram that was sent, providing that the datagram fits into the specified buffers. The excess is discarded. For stream sockets, data is not discarded. Multiple invocations of readv may be needed to return all the data.

Access Time — A successful read updates the access time of the socket read.

Number of Bytes Read — The number of bytes requested for reading is not checked against any system limit, although a limit can be imposed by a high-level-language POSIX implementation.

When a read request completes, the *return_value* field shows the number of bytes actually read — a number less than or equal to the number of bytes that were requested. The following are some reasons why the number of bytes read might be less than the number of bytes requested:

- Fewer than the requested number of bytes remained in the socket; the end of socket was reached before all requested bytes were read.
- The service was interrupted by a signal after some but not all of the requested bytes were read. (If no bytes were read, the return value is set to -1 and an error is reported.)

A read request may complete successfully with no bytes read — that is, with *return_value* set to 0. This can occur if the service specified that zero bytes are to be read.

SIGTTIN Processing — This service causes signal **SIGTTIN** to be sent if all the following conditions are met:

- The process is running in a background process group.
- The **SIGTTIN** signal is not blocked or ignored.
- The process group of the process is not orphaned.

If **SIGTTIN** has a handler, the handler gets control and the read ends with a return code of EINTR. If **SIGTTIN** is set to default, the process stops in the read and continues when the process is moved to the foreground.

Example

The following code issues a readv for a socket. SOCKDESC was returned previously from a call to either socket (BPX1SOC) or accept (BPX1ACP). This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structures, see “BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services” on page 465 and “BPXYIOV — Map the I/O Vector Structure” on page 430.

```
SPACE ,
 LA    R2,BUFFERA
 ST    R2,IOV_BASE
 LA    R2,L'BUFFERA
 ST    R2,IOV_LEN
 CALL  BPX1RDV,             Read into a vector of buffers     +
       (SOCKDESC,           Input: Socket Descriptor          +
       =A(1),               Input: Number of elements in iov  +
       IOV,                 Input: Iov containing info        +
       PRIMARYALET,         Input: Alet where iov resides     +
       PRIMARYALET,         Input: Alet of buffers for data   +
       RETVAL,              Return value: Num bytes or -1     +
       RETCODE,             Return code                       +
       RSNCODE),            Reason code                       +
       VL,MF=(E,PLIST)      ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EAGAIN	The socket is marked nonblocking, and no data is waiting to be read.
EBADF	An incorrect file descriptor was specified. The following reason codes can accompany this return code: JRFileDesNotInUse, JRFileNotOpen, JRRFileWrOnly, JRWFileRdOnly.
EINTR	A signal interrupted the readv function before any data was available.

Return Code	Explanation
EINVAL	One of the input parameters was incorrect. The following reason codes can accompany this return code: JRBytes2RWZero, JROutOfRange, JRSocketCallParmError.
EIO	The process is in a background process group and is attempting to read from its controlling terminal. However, TOSTOP is set, the process is neither ignoring nor blocking **SIGTTIN** signals, and the process group of the process is orphaned. This can happen, for example, if a background job tries to write to the terminal after the user has logged off.
ENOBUFS	A buffer could not be obtained.
ENOTSOCK	*socket_descriptor* is a valid file descriptor, but not a socket.
ESHUTDOWN	There is no data to read on the socket, and it has been shut down for reading. For AF_INET or AF_INET6, `0` is returned instead of recognizing this error.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Service

Another callable service related to this service is:

- “writev (BPX1WRV) — Write Data from a Set of Buffers” on page 404

realpath (BPX1RPH) — Find the Absolute Path Name

BPX1RPH

relative_pathname_length
relative_pathname_buffer
absolute_pathname_length
absolute_pathname_buffer
return_value
return_code
reason_code

Purpose

Use the realpath (BPX1RPH) service to determine the absolute path name for a relative path name. Any dot (.) or dot dot (..) components, symbolic links, or mount external links included in the relative path name input are resolved in the absolute path name output.

Parameters

relative_pathname_length
(input,INT,4) is a variable for specifying the length of the *relative_pathname_buffer* parameter.

relative_pathname_buffer
(input,CHAR,*relative_pathname_length*) is a variable for specifying a relative path name. See "Understanding Byte File System (BFS) Path Name Syntax" on page 6.

absolute_pathname_length
(input/output,INT,4) is a variable for specifying, on input, the length of the *absolute_pathname_buffer* parameter. If 0 is specified, the length of the buffer is assumed to be PATH_MAX plus null (1024 bytes).

On output, this parameter contains the length of the path name returned in the *absolute_pathname_buffer* parameter.

absolute_pathname_buffer
(output,CHAR,*absolute_pathname_length*) is a variable where the service returns the absolute path name.

return_value
(output,INT,4) is a variable where the service returns 0 if the request completes successfully, or -1 if the request is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Example

The following code finds the absolute path name for relative path name **../symlink1/data.file**. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551.

```
         MVC   BUFLENA,=F'21'
         MVC   BUFFERA(11),=CL21'../symlink1/data.file'
```

```
         MVC   BUFLENB,=F'0'         Buffer length = PATH_MAX + null
         SPACE ,
         CALL  BPX1RPH,              Find absolute pathname           +
               (BUFLENA,             Input: Relative pathname length  +
               BUFFERA,              Input: Relative pathname          +
               BUFLENB,              Input/output: Abs. pathname length+
               BUFFERB,              Output: Absolute pathname        +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EINVAL	One of the input parameters is not valid.
ENOENT	The BFS object does not exist.
ERANGE	The output buffer is too small to hold the absolute path name.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

recv (BPX1RCV) — Receive Data on a Socket and Store It in a Buffer

```
BPX1RCV
        socket_descriptor
        buffer_length
        buffer
        buffer_ALET
        flags
        return_value
        return_code
        reason_code
```

Purpose

Use the recv (BPX1RCV) service to receive data on a socket and store it in a buffer. If no messages are available at the socket, the service either waits for a message to arrive, or fails with the EWOULDBLOCK return code, depending on whether the socket has been defined as blocking or nonblocking.

Parameters

socket_descriptor
(input,INT,4) is a variable for specifying the descriptor of the socket.

buffer_length
(input,INT,4) is a variable for specifying the length of the *buffer* parameter.

buffer
(output,CHAR,*buffer_length*) is a variable where the service stores the received data.

buffer_ALET
(input,INT,4) is a variable for specifying the access list entry token (ALET) for *buffer*.

Note: This parameter is ignored.

flags
(input,INT,4) is a variable for specifying information about how the data is to be received. This field is mapped by the BPXYMSGF macro. See "BPXYMSGF — Map the Message Flags" on page 441.

return_value
(output,INT,4) is a variable where the service returns one of the following:

- The number of bytes received into the buffer, if the request is successful.
- `0`, indicating the connection is closed.
- `-1`, if the request is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Note

The recv callable service applies only to connected sockets. It can be used with datagram or stream sockets. For datagram sockets, the recv service returns the entire datagram that was sent, providing that the datagram fits into the specified buffers. The excess is discarded. For stream sockets, data is not discarded. Multiple invocations of the recv service may be needed to return all the data.

Example

The following code issues a recv for a socket. SOCKDESC was returned previously from a call to either socket (BPX1SOC) or accept (BPX1ACP). This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structures, see “BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services” on page 465 and “BPXYMSGF — Map the Message Flags” on page 441.

```
         SPACE ,
         CALL  BPX1RCV,              Receive data on from a socket     +
               (SOCKDESC,            Input: Socket Descriptor          +
               =A(L'BUFFERA),        Input: Length of input buffer     +
               BUFFERA,              Input: Address of input buffer    +
               PRIMARYALET,          Input: Alet of input buffer       +
               MSG_FLAGS,            Input: Flags                      +
               RETVAL,               Return value: Num bytes, 0, or -1 +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	An incorrect file descriptor was specified. The following reason codes can accompany this return code: JRFileDesNotInUse, JRFileNotOpen.
EINTR	A signal interrupted the service before any data was available.
EINVAL	The socket is marked shutdown for read.
EIO	There has been a network or transport failure. The following reason code can accompany this return code: JRPrevSockError.
ENOBUFS	A buffer could not be obtained.
ENOTCONN	A receive was attempted on a connection-oriented socket that is not connected. For AF_INET or AF_INET6, 0 is returned instead of recognizing this error.
ENOTSOCK	*socket_descriptor* does not refer to a valid socket descriptor. The following reason code can accompany this return code: JRMustBeSocket.
ESHUTDOWN	There is no data to read on the socket, and it has been shut down for reading. For AF_INET or AF_INET6, 0 is returned instead of recognizing this error.
EWOULDBLOCK	The socket is marked nonblocking, and no data is waiting to be received.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Service

Another callable service related to this service is:

- “send (BPX1SND) — Send Data on a Socket” on page 277

recvfrom (BPX1RFM) — Receive Data from a Socket and Store It in a Buffer

BPX1RFM

socket_descriptor
buffer_length
buffer
buffer_ALET
flags
sockaddr_length
sockaddr
return_value
return_code
reason_code

Purpose

Use the recvfrom (BPX1RFM) service to receive data on a socket and store it in a buffer. It can be used by an application program to receive data from sockets. When no data is available at the socket, the service either waits for data to arrive, or returns an EWOULDBLOCK return code, depending on whether the socket is defined as blocking or nonblocking.

Parameters

socket_descriptor
(input,INT,4) is a variable for specifying the descriptor of the socket.

buffer_length
(input,INT,4) is a variable for specifying the length of the *buffer* parameter.

buffer
(output,CHAR,*buffer_length*) is a variable for the buffer where the service stores the received data.

buffer_ALET
(input,INT,4) is a variable for specifying the access list entry token (ALET) for *buffer*.

Note: This parameter is ignored.

flags
(input,INT,4) is a variable for specifying information about how the data is to be received. This field is mapped by the BPXYMSGF macro. See "BPXYMSGF — Map the Message Flags" on page 441.

sockaddr_length
(input/output,INT,4) is a variable for specifying the length of the *sockaddr* parameter. This value should be large enough to accommodate the maximum length of the SOCKADDR structure to be returned in *sockaddr*, but less than 4096 bytes (4KB). On output, the service updates this field with the size of the data returned in *sockaddr*.

sockaddr
(output,INT,*sockaddr_length*) is a variable where the service returns the SOCKADDR structure containing the socket address of the sender of the data. This field is mapped by the BPXYSOCK macro. See "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465.

return_value
(output,INT,4) is a variable where the service returns the number of bytes received into the buffer if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Note

The recvfrom callable service can be used with datagram or stream sockets. For datagram sockets, it returns the entire datagram that was sent, providing that the datagram fits into the specified buffer. The excess is discarded. For stream sockets, data is not discarded. Multiple invocations of recvfrom may be needed to return all the data.

Example

The following code issues a recv from a socket. SOCKDESC was returned from a previous call to either socket (BPX1SOC) or accept (BPX1ACP). This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structures, see “BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services” on page 465 and “BPXYMSGF — Map the Message Flags” on page 441.

```
         SPACE ,
         MVC   MSG_FLAGS4,MSG_PEEK
         CALL  BPX1RFM,              Read from a socket               +
               (SOCKDESC,            Input: Socket Descriptor         +
               =A(L'BUFFERA),        Input: Length of the buffer      +
               BUFFERA,              Output: The data buffer          +
               PRIMARYALET,          Input: Alet of the buffer        +
               MSG_FLAGS,            Input: Flags                     +
               =A(L'SOCKADDR),       Input: Length of the socket addr +
               SOCKADDR,             Output: The socket address       +
               RETVAL,               Return value: Num bytes or -1    +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	An incorrect file descriptor was specified. The following reason codes can accompany this return code: JRFileDesNotInUse, JRFileNotOpen.
EINTR	A signal interrupted the recvfrom function before any data was available.
EINVAL	One of the input parameters was incorrect. The following reason code can accompany this return code: JRSocketCallParmError.
EIO	There was an I/O error. The following reason code can accompany this return code: JRPrevSockError.
ENOBUFS	A buffer could not be obtained.
ENOTCONN	A receive was attempted on a connection-oriented socket that is not connected. For AF_INET, 0 is returned instead of recognizing this error.
ENOTSOCK	*socket_descriptor* does not refer to a valid socket descriptor. The following reason code can accompany this return code: JRMustBeSocket.

Return Code	Explanation
ESHUTDOWN	There is no data to read on the socket, and it has been shut down for reading. For AF_INET or AF_INET6, 0 is returned instead of recognizing this error.
EWOULDBLOCK	The socket is marked nonblocking, and no data is waiting to be read.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Another callable service related to this service is:

- “sendto (BPX1STO) — Send Data on a Socket” on page 283

recvmsg (BPX2RMS) — Receive Messages on a Socket and Store Them in Message Buffers

BPX2RMS

socket_descriptor
message_header
flags
IOV_ALET
IOV_buffer_ALET
return_value
return_code
reason_code

Purpose

Use the recvmsg (BPX2RMS) service to receive messages on a socket and store them in a set of buffers. The socket can be either connected or unconnected. If no messages are available at the socket, the service either waits for a message to arrive or returns an EWOULDBLOCK return code, depending on whether the socket has been defined as blocking or nonblocking.

Parameters

socket_descriptor
(input,INT,4) is a variable for specifying the descriptor of the socket.

message_header
(input,CHAR,length of BPXYMSGH) is a variable for specifying the message header. This field is mapped by the BPXYMSGH macro. A message header contains a pointer to an I/O vector structure, which contains information about the buffers into which the messages are to be received, and a pointer to a SOCKADDR structure containing the socket address of the sender of the data. The I/O vector structure is mapped by the BPXYIOV macro. See "BPXYMSGH — Map the Message Headers" on page 443 and "BPXYIOV — Map the I/O Vector Structure" on page 430. The SOCKADDR structure is mapped by the BPXYSOCK macro. See "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465 for information on the BPXYSOCK macro.

flags
(input,INT,4) is a variable for specifying information about how the data is to be received. This field is mapped by the BPXYMSGF macro. See "BPXYMSGF — Map the Message Flags" on page 441.

IOV_ALET
(input,INT,4) is a variable for specifying the access list entry token (ALET) for the I/O vector (IOV) structure specified in *message_header*.

Note: This parameter is ignored.

IOV_buffer_ALET
(input,INT,4) is a variable for specifying the ALET for the buffers that are pointed to by the IOV structure in *message_header*.

Note: This parameter is ignored.

return_value
(output,INT,4) is a variable where the service returns the number of bytes read into the buffers if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Note

The BPX2RMS call supersedes the BPX1RMS call, which is still available for migration purposes only.

Example

The following code issues a recvmsg for a socket. SOCKDESC was returned from a previous call to either socket (BPX1SOC) or accept (BPX1ACP). This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structures, see "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465, "BPXYMSGF — Map the Message Flags" on page 441, "BPXYMSGH — Map the Message Headers" on page 443, and "BPXYIOV — Map the I/O Vector Structure" on page 430.

```
         SPACE ,
         XC    MSGH,MSGH             Clear msgh
         LA    R2,SOCKADDR
         ST    R2,MSGHNAMEPTR        Store the address of sockaddr
         LA    R2,SOCK#LEN+SOCK_SUN#LEN
         ST    R2,MSGHNAMELEN
         LA    R2,IOV
         ST    R2,MSGHIOVPTR
         MVI   MSGHIOVNUM,1
         LA    R2,BUFFERA
         ST    R2,IOV_BASE
         LA    R2,L'BUFFERA
         ST    R2,IOV_LEN
*
         CALL  BPX2RMS,              Receive a message from a socket   +
               (SOCKDESC,            Input: Socket Descriptor          +
               MSGH,                 Input: Address of BPXYMSGH        +
               MSG_FLAGS,            Input: Flags                      +
               PRIMARYALET,          Input: Alet of the iov            +
               PRIMARYALET,          Input: Alet of the buffers in iov +
               RETVAL,               Return value: Num bytes or -1     +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	An incorrect file descriptor was specified. The following reason codes can accompany this return code: JRFileDesNotInUse, JRFileNotOpen.
EINTR	A signal interrupted the recvmsg service before any data was available.
EINVAL	One of the input parameters was incorrect. The following reason codes can accompany this return code: JROutOfRange, JRSocketCallParmError.
EIO	There was an I/O error. The following reason code can accompany this return code: JRPrevSockError.
ENOBUFS	A buffer could not be obtained.
ENOTCONN	A receive was attempted on a connection-oriented socket that is not connected. For AF_INET, 0 is returned instead of recognizing this error.

Return Code	Explanation
ENOTSOCK	*socket_descriptor* does not refer to a valid socket descriptor. The following reason code can accompany this return code: JRMustBeSocket.
ESHUTDOWN	There is no data to read on the socket, and it has been shut down for reading. For AF_INET or AF_INET6, `0` is returned instead of recognizing this error.
EWOULDBLOCK	The socket is marked nonblocking, and no data is waiting to be read.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Service

Another callable service related to this service is:

- “sendmsg (BPX2SMS) — Send Messages on a Socket” on page 280

rename (BPX1REN) — Rename a File or Directory

BPX1REN

old_name_length
old_name
new_name_length
new_name
return_value
return_code
reason_code

Purpose

Use the rename (BPX1REN) service to change the name of a file or directory.

Parameters

old_name_length
(input,INT,4) is a variable for specifying the length of the current path name of the file or directory to be renamed.

old_name
(input,CHAR,*old_name_length*) is a variable for specifying the current path name of the file or directory.

new_name_length
(input,INT,4) is a variable for specifying the length of the new path name of the file or directory.

new_name
(input,CHAR,*new_name_length*) is a variable for specifying the new path name of the file or directory.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

The rename (BPX1REN) service changes the name of a file or directory from *old_name* to *new_name*. When renaming completes successfully, the change and modification times for the parent directories of *old_name* and *new_name* are updated.

For renaming to succeed, the calling process needs write permission for the directory containing *old_name* and the directory containing *new_name*. If *old_name* and *new_name* are the names of directories, the caller does not need write permission for the directories themselves.

Renaming Files: If *old_name* and *new_name* are links referring to the same file, rename (BPX1REN) simply returns successfully.

If *old_name* is the name of a file, *new_name* must also name a file, not a directory. If *new_name* is an existing file, it is unlinked. Then the file specified as *old_name* is renamed to *new_name*. The path name

new_name always stays in existence; at the beginning of the operation, *new_name* refers to its original file, and at the end, it refers to the file that used to be *old_name*.

Renaming Directories: If *old_name* is the name of a directory, *new_name* must also name a directory, not a file. If *new_name* is an existing directory, it must be empty, containing no files or subdirectories. If empty, it is removed, as described in "rmdir (BPX1RMD) — Remove a Directory" on page 256.

The *new_name* directory cannot be a directory under *old_name*; that is, the old directory cannot be part of the path name prefix of the new one.

Example

The following code change the directory name of a file from **usr/sam** to **usr/samantha**. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551.

```
         MVC   BUFFERB(07),=CL07'usr/sam'
         MVC   BUFLENB,=F'07'
         MVC   BUFFERA(12),=CL12'usr/samantha'
         MVC   BUFLENA,=F'12'
         SPACE ,
         CALL  BPX1REN,              Rename a file                    +
               (BUFLENB,             Input: Old name length           +
               BUFFERB,              Input: Old name                  +
               BUFLENA,              Input: New name length           +
               BUFFERA,              Input: New name                  +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The process did not have search permission on some component of the old or new path name, or did not have write permission on the parent directory of the file or directory to be renamed.
EBUSY	The *old_name* and *new_name* parameters specify directories, but one of them cannot be renamed because it is in use as a root or a mount point. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRIsFSRoot.
EINVAL	This error is returned for one of the following reasons: • The *old_name* value is part of the path name prefix of *new_name*. • The *old_name* value is either `.` (dot) or `..` (dot-dot). • The *new_name* value is either `.` (dot) or `..` (dot-dot). Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRDotOrDotDot and JROldPartOfNew.
EISDIR	The *new_name* parameter identifies a directory, but the *old_name* parameter is not a directory. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRNewIsDir.
ELOOP	A loop exists in symbolic links encountered during resolution of the *old_name* or *new_name* argument. This error is issued if more than 8 symbolic links are detected in the resolution of *old_name* or *new_name*.

Return Code	Explanation
ENAMETOOLONG	The *old_name* or *new_name* parameter is longer than 1023 bytes, or a component of those names is longer than 255 bytes. CMS does not support name truncation.
ENOENT	No file or directory named *old_name* was found, or either *old_name* or *new_name* was not specified. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JROldNoExist.
ENOSPC	The directory intended to contain *new_name* cannot be extended.
ENOTDIR	A component of either path name prefix is not a directory, or *old_name* is a directory and *new_name* is a file that is not a directory. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRNewNotDir.
ENOTEMPTY	The *new_name* parameter identifies a directory, but the directory is not empty. It contains files or subdirectories.
EROFS	Performing the requested service would make it necessary to write on a read-only file system. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRReadOnlyFS.
EXDEV	The *old_name* and *new_name* parameters identify files or directories on different file systems. CMS does not support renaming across file systems. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRDiffFileSets.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “link (BPX1LNK) — Create a Link to a File” on page 149
- “rmdir (BPX1RMD) — Remove a Directory” on page 256
- “unlink (BPX1UNL) — Remove a Directory Entry” on page 379.

rewinddir (BPX1RWD) — Reposition a Directory Stream to the Beginning

BPX1RWD

directory_file_descriptor
return_value
return_code
reason_code

Purpose

Use the rewinddir (BPX1RWD) service to "rewind", or reset to the beginning, an open directory. The next call to the readdir (BPX1RDD) service reads the first entry in the directory.

Parameters

directory_file_descriptor
(input,INT,4) is a variable for specifying the directory file descriptor of the directory to be "rewound". This value was returned by the opendir (BPX1OPD) service when the directory was opened.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Note

If the contents of the directory you specify have changed since the directory was opened, a call to the rewinddir (BPX1RWD) service updates the directory and a subsequent call to the read service reads the new contents.

Example

The following code resets the open directory to the beginning. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   DIRECTDES,..          File descriptor from opendir
         CALL  BPX1RWD,              Reposition directory at beginning +
               (DIRECTDES,           Input: Directory file descriptor  +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return code:

Return Code	Explanation
EBADF	The *directory_file_descriptor* parameter does not represent an open directory. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRRwdFileNotDir.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “closedir (BPX1CLD) — Close a Directory” on page 36
- “opendir (BPX1OPD) — Open a Directory” on page 185
- “readdir (BPX1RDD) — Read an Entry from a Directory” on page 231.

rmdir (BPX1RMD) — Remove a Directory

BPX1RMD

directory_name_length

directory_name

return_value

return_code

reason_code

Purpose

Use the rmdir (BPX1RMD) service to remove a directory. The directory must be empty.

Parameters

directory_name_length
(input,INT,4) is a variable for specifying the length of the *directory_name* parameter.

directory_name
(input,CHAR,*directory_name_length*) is a variable for specifying the path name of the directory to be removed.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. The directory must be empty.
2. If the directory is successfully removed, the change and modification times for the parent directory are updated.
3. If the link count of the directory becomes zero and no process has the directory open, the directory itself is deleted. The space occupied by the directory is freed for new use and the contents of the file are lost.
4. If any process has the directory open when the last link is removed, the directory itself is not removed until the last process closes the directory. New files cannot be created under a directory after the last link is removed, even if the directory is still open.

Example

The following code removes directory **applib/user02**. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   BUFFERA(13),=CL13'applib/user02'
         MVC   BUFLENA,=F'13'
         SPACE ,
         CALL  BPX1RMD,             Remove a directory               +
               (BUFLENA,            Input: Directory name length     +
               BUFFERA,             Input: Directory to be removed   +
```

```
                RETVAL,              Return value: 0 or -1             +
                RETCODE,             Return code                       +
                RSNCODE),            Reason code                       +
                VL,MF=(E,PLIST)      ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The process did not have search permission for some component of *directory_name*, or did not have write permission for the directory containing the directory to be removed.
EBUSY	The directory cannot be removed, because it is being used by a process. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRRootNode.
ECMSERR	An internal error occurred.
EINVAL	The argument supplied was incorrect. Examples of incorrect arguments are `.` (dot) and `..` (dot-dot). Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRDotOrDotDot.
ELOOP	A loop exists in symbolic links encountered during resolution of the *directory_name* argument. This error is issued if more than 8 symbolic links are detected in the resolution of *directory_name*.
ENAMETOOLONG	The name of the directory is longer than 1023 characters, or some component of the path name is longer than 255 characters. This could be as a result of encountering a symbolic link during resolution of *directory_name*, and the substituted string is longer than 1023 characters.
ENOENT	The directory specified by *directory_name* was not found, or no *directory_name* parameter was specified. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRFileNotThere.
ENOTDIR	Some component of *directory_name* is not a directory. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRPathNotDir.
ENOTEMPTY	The directory contains files or subdirectories.
EROFS	The directory to be removed is on a read-only file system. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRReadOnlyFS.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “mkdir (BPX1MKD) — Make a Directory” on page 160
- “unlink (BPX1UNL) — Remove a Directory Entry” on page 379.

select/selectex (BPX1SEL) — Select on File Descriptors and Message Queues

BPX1SEL

msgfd_count
read_list_length
read_list
write_list_length
write_list
exception_list_length
exception_list
timeout_pointer
ECB_pointer
user_option_field
return_value
return_code
reason_code

Purpose

Use the select/selectex (BPX1SEL) service to check the I/O status of multiple open file descriptors and message queues. The file descriptors can be for character special files, pipes, sockets, or files.

Parameters

msgfd_count
(input,INT,4) is a variable for specifying the number of items to be checked. The first halfword (the high-order 16 bits) indicates the number of message queues. The second halfword (the low-order 16 bits) indicates the number of file descriptors.

The number of message queues indicates the number of elements (queue IDs) in each of the arrays contained in *read_list*, *write_list*, and *exception_list*. For example, if you specify a value of 10 in the first halfword of *msgfd_count*, it is expected that *read_list*, *write_list*, and *exception_list* each contain an array of 10 elements. If you specify a value of 0, it is assumed that no arrays are given and no message queues are to be checked. The maximum number of message queues that you can specify is 32 767.

The number of file descriptors is the highest file descriptor that is being checked for status, plus 1. For example, if you are interested in the I/O status of file descriptors 5 and 8, the second halfword of *msgfd_count* should be 9. (Numbering of file descriptors begins with 0, so fd 8 is actually the 9th file descriptor.) If you want to check file descriptors for status along with message queues, the highest file descriptor you can specify is 2046.

read_list_length
(input,INT,4) is a variable for specifying the length of the *read_list* parameter. This length is the sum of the length of the bit set specifying file descriptors, rounded up to a multiple of 4 bytes, and the length of the array of message queue identifiers. When both file descriptors and message queues are specified, this field should contain a value greater than 256 bytes. If 0 is specified, the *read_list* is not checked by the service. The value can be in the range from 0 to 5000.

read_list
(input/output,CHAR,*read_list_length*) is a variable for specifying a structure that contains the bit set for the specified file descriptors and the array of message queue identifiers.

The bit set must be padded with extra bytes, if necessary, to round up its length to the next multiple of 4 bytes. The bits in the bit set should be turned on for the corresponding descriptors to be checked for reading. The format of the bits can be specified with the *user_option_field* parameter. On return, the service sets the bits for those descriptors that are ready for reading.

If *read_list* contains both a bit set and an array of message queue identifiers, the bit set must be 256 bytes in length. If only file descriptors are to be checked, the bit set can have any valid size.

Each element of the array of message queue identifiers is 4 bytes in length. Elements with a value of -1 are acceptable and are ignored. On return, the service replaces message queue identifiers that do not meet the criterion with a value of -1.

write_list_length
(input,INT,4) is a variable for specifying the length of the *write_list* parameter. This length is the sum of the length of the bit set specifying file descriptors, rounded up to a multiple of 4 bytes, and the length of the array of message queue identifiers. When both file descriptors and message queues are specified, this field should contain a value greater than 256 bytes. If 0 is specified, the *write_list* is not checked by the service. The value can be in the range from 0 to 5000.

write_list
(input/output,CHAR,*write_list_length*) is a variable for specifying a structure that contains the bit set for the specified file descriptors and the array of message queue identifiers.

The bit set must be padded with extra bytes, if necessary, to round up its length to the next multiple of 4 bytes. The bits in the bit set should be turned on for the corresponding descriptors to be checked for writing. The format of the bits can be specified with the *user_option_field* parameter. On return, the service sets the bits for those descriptors that are ready for writing.

If *write_list* contains both a bit set and an array of message queue identifiers, the bit set must be 256 bytes in length. If only file descriptors are to be checked, the bit set can have any valid size.

Each element of the array of message queue identifiers is 4 bytes in length. Elements with a value of -1 are acceptable and are ignored. On return, the service replaces message queue identifiers that do not meet the criterion with a value of -1.

exception_list_length
(input,INT,4) is a variable for specifying the length of the *exception_list* parameter. This length is the sum of the length of the bit set specifying file descriptors, rounded up to a multiple of 4 bytes, and the length of the array of message queue identifiers. When both file descriptors and message queues are specified, this field should contain a value greater than 256 bytes. If 0 is specified, the *exception_list* is not checked by the service. The value can be in the range from 0 to 5000.

exception_list
(input/output,CHAR,*exception_list_length*) is a variable for specifying a structure that contains the bit set for the specified file descriptors and the array of message queue identifiers.

The bit set must be padded with extra bytes, if necessary, to round up its length to the next multiple of 4 bytes. The bits in the bit set should be turned on for the corresponding descriptors to be checked for exceptions. The format of the bits can be specified with the *user_option_field* parameter. On return, the service sets the bits for those descriptors that have had exceptions.

If *exception_list* contains both a bit set and an array of message queue identifiers, the bit set must be 256 bytes in length. If only file descriptors are to be checked, the bit set can have any valid size.

Each element of the array of message queue identifiers is 4 bytes in length. Elements with a value of -1 are acceptable and will be ignored. On return, the service replaces message queue identifiers that do not meet the criterion with a value of -1.

timeout_pointer
(input,PTR,4) is a variable for specifying a pointer to a timeout value that controls how the file descriptors are checked:

0

Wait indefinitely. If the pointer is zero, the service waits (indefinitely) until one of the selected descriptors is ready.

>0

Wait for a specified period of time. If *timeout_pointer* is greater than zero, it points to the location of the timeout value. The service waits the amount of time specified in the timeout value for one of the conditions to occur before returning to the caller. The timeout value is mapped by the BPXYSELT macro (see "BPXYSELT — Map the Timeout Value for the select/selectex Service" on page 458) and consists of two fields, seconds and microseconds:

- Microseconds can be a value in the range from 0 to 1,000,000. (1,000,000 microseconds equal 1 second.)
- Seconds can be a value in the range from 0 to 2,147,483. (2,147,483 seconds equal approximately 24.85 days.)

Notes:

1. Microseconds and seconds are added together to determine the timeout value.
2. If the timeout value is more than 0 and less than 300 microseconds, the value is rounded up to 300 microseconds.
3. The maximum time that can be specified is 2,147,483 seconds and 647,000 microseconds (2^{31}-1 microseconds).
4. A timeout value of **0** means **Do not wait.** The service returns immediately after checking the selected descriptors; no waiting is done.

ECB_pointer

(input,PRT,4) is a variable for specifying one of the following:

- A pointer to a user event control block (ECB). The high-order bit in *ECB_pointer* must be set to B'0'.
- A pointer to a list of ECBs. The high-order bit in *ECB_pointer* must be set to B'1'.
- 0, indicating no ECBs are specified.

user_option_field

(input/output,INT,4) is a variable for specifying the format of the read, write, and exception bit lists.

On input, specify one of the following (the values are defined in the BPXYSEL macro; "BPXYSEL — Map Options for the select/selectex Service" on page 456):

- SEL#BITSBACKWARD – Bit-backward order by word

  Bits are read from right to left within each word, with the low-order bit on the right and the high-order bit on the left. For example:

```
 Word 1              Word 2                 Word 3
------------------  ----------------------  ----------------------
31 30 29...3 2 1 0   63 62 61...35 34 33 32   95 94 93...67 66 65 64
------------------  ----------------------  ----------------------
```

  Note: In this example, file descriptor 0 is represented by the last bit on the right in Word 1.

- SEL#BITSFORWARD – Bit-forward order by word

  Bits are read from left to right within each word, with the low-order bit on the left and the high-order bit on the right. For example:

```
 Word 1              Word 2                 Word 3
------------------  ----------------------  ----------------------
0 1 2 3...29 30 31   32 33 34 35...61 62 63   64 65 66.67...93 94 95
------------------  ----------------------  ----------------------
```

  Note: In this example, file descriptor 0 is represented by the first bit on the left in Word 1.

On output, the service returns one of the following:

- -1, indicating that all the selected file descriptors supported the select service.
- The first selected file descriptor that did not support the select service.

return_value
(output,INT,4) is a variable where the service returns one of the following:

- The number of read, write, and exceptional conditions that were found among the specified message queues and file descriptors. The first halfword indicates the number of exceptions in the messages queues; the second halfword indicates the number of exceptions in the file descriptors. If the value for the message queues exceeds 32 767, only 32 767 is reported. This is to ensure that *return_value* does not appear to be negative. Should the value for the file descriptors be greater than 65 535, only 65 535 is reported.
- 0, if the timeout value expired before any of the conditions were met.
- -1, if the request is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. The bit set for the *read_list*, *write_list*, and *exception_list* is a string of bits such that if X is an element of the set, the bit that represents X is set to 1. For example, if descriptor 1 is to be checked, bit 1 should be turned on in the bit set. Here is how that byte would look:
   - Bit-forward order: B'01000000'.
   - Bit-backward order: B'00000010'.
2. When a positive value is specified for the number of file descriptors:
   - At least one bit set (read, write, or exception) must be specified, and its length must be large enough (rounded up to the next multiple of 4) to contain the bit that represents the largest descriptor you specified.
   - If more than one bit set is specified, each bit set must be the same length.

     For example, if you want to check the read status for file descriptor 59 and the write status for file descriptor 6:

     a. Number of fds = 60 (the largest fd plus 1)
     b. *read_list_length* = 8
     c. *read_list* = the bit representing fd 59 is set on (see *user_option_field* to determine which bit that would be)
     d. *write_list_length* = 8
     e. *write_list* = the bit representing fd 6 is set on (see *user_option_field* to determine which bit that would be)
     f. *exception_list_length* = 0
3. When both the first and second halfwords of *msgsfd_count* contain a positive value, *read_list*, *write_list*, and *exception_list* must each contain both a bit set and an array of message queue identifiers, unless a value of 0 is specified for its length. The following example illustrates what you must do.

   Suppose you want to check the read status for file descriptors 3 and 5 and the write status for message queues whose identifiers are 7 and 8:

   a. Number of fds = 6 (the largest fd plus 1)

b. Number of message queues = 2

c. *read_list_length* = 264 (256 byte bit set length + 8 byte array length)

d. *read_list* = the 256-byte bit set with appropriate bits set on for fds 3 and 5, followed by a two-element array that contains the value of -1 in both elements.

e. *write_list_length* = 264 (same length as for read)

f. *write_list* = the 256-byte bit set with all its bits set off followed by the two-element array that contains the numbers 7 and 8.

g. *exception_list_length* = 0

4. You can use the select service as a timer-only function by specifying zero for either or both of the following:

   - *msgfd_count*
   - *read_list_length*, *write_list_length*, and *exception_list_length*

   and by specifying *timeout_pointer* and a timeout value. If you specify zero for *timeout_pointer*, the select service blocks forever. If you specify a timeout value of zero, no blocking is done, and the select service returns immediately to the caller.

5. You can also specify *ECB_pointer* with the timer-only function.
6. Regular files are always ready for reading and writing.

Example

The following code issues a select for a previously connected socket. SOCKDESC was returned when the socket was created. In this case, the select is for a single socket for read, write and exception. Do not request waiting. There are no ECBs. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structures, see “BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services” on page 465 and “BPXYSEL — Map Options for the select/selectex Service” on page 456.

```
         SPACE ,
*
         MVC   SELLIST(4),=XL4'80000000'                            +
                                  Turn on the bit representing sd 0
*
         CALL  BPX1SEL,              Select on a set of sockets     +
               (SOCKDESC+1,          Input: Number of file descriptors +
               =A(4),                Input: Length of read list     +
               SELLIST,              Input: Address of read list    +
               =A(4),                Input: Length of write list    +
               SELLIST,              Input: Address of write list   +
               =A(4),                Input: Length of exception list +
               SELLIST,              Input: Address of exception list +
               =A(0),                Input: Timeout value           +
               =A(0),                Input: ECB pointer             +
               =A(SEL#BITSFORWARD),  Input: Option - bits forward   +
               RETVAL,               Return value: Num found, 0, or -1 +
               RETCODE,              Return code                    +
               RSNCODE),             Reason code                    +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return_code	Explanation
ECANCELED	The asynchronous I/O request was canceled. The following reason code can accompany this return code: JREcbError.
ECMSERR	A CMS environmental or internal error has occurred. The following reason code can accompany this return code: JRInternalError.

Return_code	Explanation
ECMSSTORAGE	There was a storage management error. The following reason codes can accompany this return code: JRStorageObtainErr, JRStorageReleaseErr.
EINTR	The select service request was interrupted by a signal for the caller.
EINVAL	One of the input parameters was not correct. The following reason codes can accompany this return code: JREcbError2, JRInvUserOp, JRListLenBad, JRListTooShort, JRMsOutOfRange, JRNoFdsTooManyQIds, JRNoLists, JRSecOutOfRange, JRTooManyFds.
EIO	There was an I/O error.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

semctl (BPX1SCT) — Perform Semaphore Control Operations

BPX1SCT

semaphore_set_ID
semaphore_number
command
value_or_address
return_value
return_code
reason_code

Purpose

Use the semctl (BPX1SCT) service to do various semaphore control operations, including getting status, changing variables, and removing a semaphore set from the system.

Parameters

semaphore_set_ID
(input,INT,4) is a variable for specifying the semaphore set identifier. This value is returned by the semget (BPX1SGT) service.

semaphore_number
(input,INT,4) is a variable for specifying the number of a particular semaphore in *semaphore_set_ID*. This value can range from zero to one less than the number of semaphores in the semaphore set. Use this parameter with the SEM_GETVAL, SEM_SETVAL, SEM_GETNCNT, or SEM_GETZCNT command. The parameter is ignored for all other commands.

command
(input,INT,4) is a variable for specifying a command that identifies the operation to be performed. The SEM_ command constants are defined in the BPXYSEM macro. The IPC_ command constants are defined in the BPXYIPCP macro. See "BPXYSEM — Map Interprocess Communications Semaphores" on page 459 and "BPXYIPCP — Map Interprocess Communications Permissions" on page 431. The possible commands are:

SEM_GETVAL
Gets the value of *semval* for the specified *semaphore_number*. The current process must have read permission.

SEM_SETVAL
Sets the value of *semval* for the specified *semaphore_number* to the value contained in the *value_or_address* parameter. The current process must have alter permission.

When the service completes successfully, the *semadj* values corresponding to the specified *semaphore_number* for all processes are cleared.

SEM_GETPID
Gets the process ID of the most recent process to update the specified *semaphore_number*. The current process must have read permission.

SEM_GETNCNT
Gets the number of threads waiting for the *semval* of the specified *semaphore_number* to become greater than the current value. The current process must have read permission.

SEM_GETZCNT
Gets the number of threads waiting for the *semval* of the specified *semaphore_number* to become zero. The current process must have read permission.

SEM_GETALL
Gets the *semval* for all the semaphores in *semaphore_set_ID* and stores them into the array of halfwords pointed to by the address contained in the *value_or_address* parameter. The current process must have read permission.

It is the responsibility of the caller to ensure that the storage allocated for the array is large enough to hold all the semaphore elements. The number of semaphore values stored into the array is equal to the value contained in the SEM_NSEMS field of the SEMID_DS data structure in the BPXYSEM macro.

SEM_SETALL
Sets the *semval* for all the semaphores in *semaphore_set_ID*, according to the values contained in the array pointed to by the *value_or_address* parameter. The current process must have alter permission. Each value specified in the array must be either zero or positive. When this command is successfully executed, the *semadj* values corresponding to each of the semaphores in this semaphore set in all processes are cleared.

It is the responsibility of the caller to ensure that the storage allocated for the array is large enough for all the semaphore elements. The number of semaphore values read from the array is equal to the value contained in the SEM_NSEMS field of the SEMID_DS data structure in the BPXYSEM macro.

IPC_STAT
Obtains status information about *semaphore_set_ID*. The current process must have read permission. This information is stored in the buffer pointed to by the *value_or_address* parameter and mapped by the SEMID_DS data structure in the BPXYSEM macro.

IPC_SET
Sets the values of IPC_UID, IPC_GID, and IPC_MODE for *semaphore_set_ID*. The values to be set are taken from the SEMID_DS data structure in the buffer pointed to by the *value_or_address* parameter. You can specify any value for IPC_UID and IPC_GID. For IPC_MODE, you can specify only the mode bits defined for the *semaphore_flag*s parameter of the semget (BPX1SGT) service.

Note: The IPC_ values set with this command are defined in the BPXYIPCP macro and mapped into the SEM_PERM field of the SEMID_DS structure in the BPXYSEM macro. In addition, the IPC_MODE field in BPXYIPCP is mapped by the BPXYMODE macro.

IPC_RMID
Removes *semaphore_set_ID* from the system. This operation removes the identifier and destroys the set of semaphores and the SEMID_DS data structure associated with it.

The IPC_SET and IPC_RMID operations can be performed only by a process that has either appropriate privileges or an effective user ID equal to the value of IPC_CUID or IPC_UID in the SEMID_DS data structure associated with *semaphore_set_ID*.

For the SEMID_DS data structure, see "BPXYSEM — Map Interprocess Communications Semaphores" on page 459.

value_or_address
(input,INT,4) is a variable for specifying a value, an address, or a null, depending on the specified *command*. Table 3 on page 265 shows the relationship of the *semaphore_number*, *command*, *value_or_address*, and *return_value* parameters. (The return value shown is for successful completion.) A dash "—" in the table means the parameter is ignored.

Table 3. Contents of value_or_address Parameter

semaphore_number	***command***	***value_or_address***	***return_value***
semaphore number	GETVAL	—	current *semval*
semaphore number	SETVAL	new *semval*	0
semaphore number	GETPID	—	last *sempid*
semaphore number	GETNCNT	—	*semncnt*

Table 3. Contents of value_or_address Parameter (continued)

semaphore_number	***command***	***value_or_address***	***return_value***
semaphore number	GETZCNT	—	*semzcnt*
—	GETALL	address of array	0
—	SETALL	address of array	0
—	STAT	address of buffer	0
—	SET	address of buffer	0
—	RMID	—	0

return_value
(output,INT,4) is a variable where the service returns a value or 0 (see Table 3 on page 265) if the request is successful, or -1 if it is unsuccessful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. Each semaphore in the semaphore set is represented by an anonymous data structure defined as follows:

   semval
   unsigned halfword semaphore value

   sempid
   process ID of the last operation

   semncnt
   unsigned halfword number of processes waiting for *semval* to become greater than the current value

   semzcnt
   unsigned halfword number of processes waiting for *semval* to become zero
2. A *semadj* variable is maintained by the process for all of its threads. This adjustment value allows the kernel to restore semaphore values in the event a process terminates before it can issue a semop (BPX1SOP) call. Maintaining *semadj* values for process termination is the application's responsibility.
3. The IPC_SET operation can change permissions, which may affect the ability of a thread to use the semaphore callable services.
4. When an IPC_RMID command is processed, all waiting threads regain control with a return value of -1, a return code of EIDRM, and a reason code of JRIpcRemoved.
5. For an IPC_RMID operation, the removal of the semaphore set will be complete by the time control is returned to the caller.

Characteristics and Restrictions

The invoker is restricted by the ownership, read, and read-write permissions defined by the semget (BPX1SGT) and semctl (BPX1SCT) services.

Example

The following code retrieves the PID of the last process to update semaphore 4 from the SEM_ID semaphore set. For the data structure, see "BPXYSEM — Map Interprocess Communications Semaphores" on page 459.

```
         LA    R15,BUFFERA
         ST    R15,BUFA
         MVC   SEM_NUMBER(4),4       Semaphore number 4 in set
         SPACE ,
         CALL  BPX1SCT,              Semaphore control operations      +
               (SEM_ID,              Input: Semaphore set ID           +
               SEM_NUMBER,           Input: Semaphore number (0 based) +
               =A(SEM_GETPID),       Input: Action to take     BPXYSEM +
               BUFA,                 Input: Value | Buffer | Array | 0 +
               RETVAL,               Return value: 0, -1 or value      +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	Operation permission (read or alter) is denied to the calling process. The following reason code can accompany this return code: JRIpcDenied.
EFAULT	The *value_or_address* parameter specified an address that caused the service to program check. The following reason code can accompany this return code: JRBadAddress.
EINVAL	One of the following conditions is true: • *semaphore_set_ID* is not a valid semaphore set identifier. • *semaphore_number* is less than zero, or greater than or equal to the number of semaphores in this set. • *command* is not a valid command. • The mode bits set by the IPC_SET command were not valid. The following reason codes can accompany this return code: JRIpcBadFlags, JRIpcBadID, JRSema4BadSemN, JRBadEntryCode.
EPERM	The IPC_SET or IPC_RMID command was specified, but the caller has neither appropriate privileges nor an effective user ID equal to the value of IPC_CUID or IPC_UID in the SEMID_DS data structure associated with *semaphore_set_ID*. The following reason code can accompany this return code: JRIpcDenied.
ERANGE	The value specified in the *value_or_address* parameter with the SEM_SETVAL or SEM_SETALL command exceeds the system-imposed maximum defined by SEM#MAX_VAL in the BPXYSEM macro. The following reason code can accompany this return code: JRSema4BadValue.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, "Return Codes," on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, "Reason Codes," on page 495.

Related Services

Other callable services related to this service are:

- “cmsprocclp (BPX1MPC) — Clean Up Kernel Resources” on page 38
- “semget (BPX1SGT) — Create or Find a Set of Semaphores” on page 269
- “semop (BPX1SOP) — Perform Semaphore Serialization Operations” on page 273

semget (BPX1SGT) — Create or Find a Set of Semaphores

BPX1SGT

key
number_of_semaphores
semaphore_flags
return_value
return_code
reason_code

Purpose

Use the semget (BPX1SGT) service to create a new semaphore set or find an existing semaphore set (if the user is allowed to access it). The service returns a system-assigned semaphore set identifier.

Parameters

key
(input,INT,4) is a variable for specifying a user-defined value that identifies a semaphore set. The *key* serves as a lookup value to determine if an associated semaphore set identifier already exists. If an associated semaphore set identifier does not already exist, the *key* value becomes associated with the semaphore set identifier created by this request.

The reserved key value IPC_PRIVATE may also be specified. IPC_PRIVATE is sometimes used when a process does not want to share a semaphore set or when it wants to privately control access to the semaphore set by other processes. The IPC_PRIVATE constant is defined in the BPXYIPCP macro. See "BPXYIPCP — Map Interprocess Communications Permissions" on page 431.

number_of_semaphores
(input,INT,4) is a variable for specifying the number of semaphores to be allocated to this set. The maximum value for this variable is controlled by the installation. If the application knows that the semaphore set associated with *key* already exists, a value of zero may be specified; this value must not be greater than the number of semaphores in the existing set. A value of zero is not allowed with the IPC_PRIVATE key or the IPC_CREAT flag.

semaphore_flags
(input,INT,4) is a variable for specifying the type of action to be performed and the permissions to be assigned. Valid values for this parameter include any combination of the following flags (additional bits will cause an EINVAL return code):

- These flags are defined in the BPXYIPCP macro and the values are mapped onto the S_TYPE field in the BPXYMODE macro:

  IPC_CREAT
  Creates a semaphore set if the specified *key* is not associated with a semaphore set identifier. IPC_CREAT is ignored when the IPC_PRIVATE reserved key is specified.

  IPC_EXCL
  Causes the service to fail if the specified *key* has an associated semaphore set identifier. IPC_EXCL is ignored when the IPC_PRIVATE reserved key is specified or the IPC_CREAT flag is not set.

- These flags are defined in the BPXYMODE macro and are a subset of the access permissions that apply to files:

  S_IRUSR
  Permits the process that owns the semaphore set to read it.

S_IWUSR
Permits the process that owns the semaphore set to alter it.

S_IRGRP
Permits the group associated with the semaphore set to read it.

S_IWGRP
Permits the group associated with the semaphore set to alter it.

S_IROTH
Permits others to read the semaphore set.

S_IWOTH
Permits others to alter the semaphore set.

See "BPXYIPCP — Map Interprocess Communications Permissions" on page 431 and "BPXYMODE — Map Mode Constants" on page 437.

return_value
(output,INT,4) is a variable where the service returns the semaphore set identifier associated with *key* if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. Each semaphore in the semaphore set is represented by an anonymous data structure defined as follows:

   semval
   unsigned halfword semaphore value

   sempid
   process ID of the last operation

   semncnt
   unsigned halfword number of processes waiting for *semval* to become greater than the current value

   semzcnt
   unsigned halfword number of processes waiting for *semval* to become zero
2. When a semaphore set is created, the value of *semval* for all semaphores is set to zero.
3. As long as a thread knows the semaphore set identifier and access is permitted, the thread can issue semctl (BPX1SCT) or semop (BPX1SOP) calls for that semaphore set, and semget is not needed.
4. This service creates a data structure defined by SEMID_DS and an array containing the number of semaphores specified, if either of the following is true:
   - IPC_PRIVATE is specified in the *key* parameter.
   - The IPC_CREAT flag is set, and the specified *key* value does not already have a semaphore set identifier associated with it.

   The SEMID_DS data structure is defined in the BPXYSEM macro, and some values are mapped into it from the BPXYIPCP macro. See "BPXYSEM — Map Interprocess Communications Semaphores" on page 459 and "BPXYIPCP — Map Interprocess Communications Permissions" on page 431.
5. Upon creation, the SEMID_DS data structure is initialized as follows:
   - IPC_CUID and IPC_UID are set to the effective user ID of the calling process.
   - IPC_CGID and IPC_GID are set to the effective group ID of the calling process.

- The low-order 9-bits of IPC_MODE are equal to the low-order 9-bits of the *semaphore_flags* parameter.
- SEM_NSEMS is set equal to the value of the *number_of_semaphores* parameter.
- SEM_OTIME is set to 0 and SEM_CTIME is set to the current time.

6. If the *key* parameter is not IPC_PRIVATE, and the IPC_EXCL flag is not set, and a semaphore set identifier already exists for the specified *key*, the value of the *number_of_semaphores* parameter may not exceed the number of semaphores specified on the semget request that created the semaphore set.
7. The semaphore set is removed from the system by calling the semctl (BPX1SCT) service with the IPC_RMID command.
8. Users of semaphore sets are responsible for removing them when they are no longer needed. Failure to do so will tie up system resources.

Characteristics and Restrictions

There is a maximum number of semaphore sets and semaphores allowed in the system.

The invoker is restricted by the ownership, read, and read-write permissions for the specified semaphore set as defined by the semget (BPX1SGT) and semctl (BPX1SCT) services.

Example

The following code creates a private set of 10 semaphores. For the data structure, see "BPXYSEM — Map Interprocess Communications Semaphores" on page 459.

```
         MVC   KEY(4),=A(IPC_PRIVATE)  Local to this family
         MVI   S_TYPE,IPC_CREAT+IPC_EXCL   Must not already exist
         MVI   S_MODE1,0             Not used
         MVI   S_MODE2,S_IRUSR       All read and write permissions
         MVI   S_MODE3,S_IWUSR+S_IRGRP+S_IWGRP+S_IROTH+S_IWOTH
         MVC   NUMB_SEMS(4),=A(10)         10 semaphores this set
         SPACE ,
         CALL  BPX1SGT,              Create a set of semaphores        +
               (KEY,                 Input: Semaphore key              +
               NUMB_SEMS,            Input: Number semaphores in set   +
               S_MODE,               Input: Flags    BPXYMODE/BPXYIPCP +
               RETVAL,               Return value: -1 or Semaphore ID  +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
         SPACE ,
         ICM   R15,B'1111',RETVAL    Test return value
         BNP   PSEUDO                Branch on semget failure
         ST    R15,SEM_ID            Store SEM_ID associated with key
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	A semaphore set identifier exists for the specified *key*, but access permission, as specified by the low-order 9-bits of the *semaphore_flags* parameter (the S_ flags) is not granted. The following reason code can accompany this return code: JRIpcDenied.
EEXIST	A semaphore set identifier exists for the specified *key*, and the IPC_CREAT and IPC_EXCL flags are both set. The following reason code can accompany this return code: JRIpcExists.

Return Code	Explanation
EINVAL	One or more of the following conditions exist: • *number_of_semaphores* is not valid because: – A semaphore set identifier exists for the specified *key*, and *number_of_semaphores* exceeds the number of semaphores previously defined. – *number_of_semaphores* is zero. – *number_of_semaphores* exceeds the system limit. • *semaphore_flags* includes bits not supported by this service. The following reason codes can accompany this return code: JRSema4BadNSems, JRSema4ZeroNSems, JRSema4BigNSems, JRIpcBadFlags.
ENOENT	A semaphore set identifier does not exist for the specified *key*, and the IPC_CREAT flag is not set. The following reason code can accompany this return code: JRIpcNoExists.
ENOSPC	A semaphore set is to be created, but the system-imposed limit on the maximum number of semaphore set identifiers allocated system-wide would be exceeded. The following reason code can accompany this return code: JRIpcMaxIDs.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “w_getipc (BPX1GET) — Query Interprocess Communications” on page 391
- “semctl (BPX1SCT) — Perform Semaphore Control Operations” on page 264
- “semop (BPX1SOP) — Perform Semaphore Serialization Operations” on page 273

semop (BPX1SOP) — Perform Semaphore Serialization Operations

BPX1SOP

semaphore_set_ID
semaphore_operations
number_of_semaphore_operations
return_value
return_code
reason_code

Purpose

Use the semop (BPX1SOP) service to perform a group of semaphore operations atomically.

Parameters

semaphore_set_ID
(input,INT,2) is a variable for specifying the semaphore set identifier.

semaphore_operations
(input,INT,4) is a variable for specifying the address of an array of data structures mapped by SEM_BUF_ELE in the BPXYSEM macro. See "BPXYSEM — Map Interprocess Communications Semaphores" on page 459. Each SEM_BUF_ELE element contains the following:

SEM_NUM
This is a halfword semaphore number in the set identified by *semaphore_set_ID*. References to the *semval*, *sempid*, *semncnt*, and *semzcnt* values are to this element in the semaphore set. (See usage note "1" on page 274 for definitions of these terms.) SEM_NUM can range from 0 to *number_of_semaphores* - 1.

SEM_OP
This is a signed halfword with three different operations for modifying the *semval* for the semaphore identified by SEM_NUM:

SEM_OP < 0
Evaluates *semval* + SEM_OP (remember that SEM_OP is negative in this case). If the operation yields a negative number, the operation either returns to the caller (EAGAIN) or suspends execution of the calling thread until the operation yields a non-negative number. The *semncnt* will be incremented for each thread waiting and decremented when waiting is complete. When complete, *semval* = *semval* + SEM_OP.

SEM_OP > 0
Sets *semval* = *semval* + SEM_OP.

SEM_OP = 0
Tests the *semval*. If it is not zero, the operation either returns to the caller (EAGAIN) or suspends execution of the calling thread until *semval*=0. The *semzcnt* will be incremented for each thread waiting and decremented when waiting is complete.

All updates to the *semval* for all of the semaphores in the set are made atomically when this service completes successfully. Partial updates to the *semval* are not performed.

SEM_FLGS
This field contains the IPC_NOWAIT and SEM_UNDO bits. IPC_NOWAIT causes SEM_OP=0 and SEM_OP<0 to return immediately with a return code of EAGAIN if the condition cannot be met; otherwise, processing is suspended. SEM_UNDO instructs the process to maintain an adjustment value for SEM_OP ¬= 0.

number_of_semaphore_operations
(input,INT,4) is a variable for specifying the number of SEM_BUF_ELE elements in the array located at *semaphore_operations*. A value of zero up to the maximum allowed by the system may be specified.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful (all of the SEM_OP operations were performed), or `-1` if it is unsuccessful (none of the SEM_OP operations were performed).

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. Each semaphore in the semaphore set is represented by an anonymous data structure defined as follows:

   semval
   unsigned halfword semaphore value

   sempid
   process ID of the last operation

   semncnt
   unsigned halfword number of processes waiting for *semval* to become greater than the current value

   semzcnt
   unsigned halfword number of processes waiting for *semval* to become zero
2. A nonzero SEM_OP value requires write permission (or an EACCES return code results).
3. A zero SEM_OP value requires read permission (or an EACCES return code results).
4. Upon successful completion, *sempid* will equal the process ID of the calling process.
5. Waiters will be served on a FIFO basis.
6. Waiting is done on a thread basis. Multiple threads (even within a single process) could be waiting on the same semaphore.
7. Adjustments are maintained on a process basis and can be changed by threads outside or within the process.
8. Within an array of semaphore operations, either all of the operations or none of the operations will be performed.
9. Incorrect usage of semaphores may result in the application being deadlocked and waiting forever. Techniques such as designing semaphore hierarchy so that the semaphores are obtained in a specific order will avoid deadlocks.
10. If the *number_of_semaphore_operations* is zero, the service returns successfully with no semaphore operation being performed.

Characteristics and Restrictions

The invoker is restricted by ownership, read, and read-write permissions defined by the semget (BPX1SGT) and semctl (BPX1SCT) services.

Example

The following code retrieves the PID of the last process to update semaphore 4 from the SEM_ID semaphore set. For the data structure, see "BPXYSEM — Map Interprocess Communications Semaphores" on page 459.

```
         LA    R5,BUFFERA            ->Utility buffer
         ST    R5,BUFA
         USING SEM_BUF_ELE,R5        ->1st SEM_BUF_ELE
         MVC   SEM_NUM(2),=AL2(0)    Semaphore number 0
         MVC   SEM_OP(2),=AL2(-1)              take the resource
         MVC   SEM_FLG(2),=AL2(SEM_UNDO)       flags (undo,wait)
         LA    R5,SEM#BUFLEN(,R5)    ->next SEM_BUF_ELE
         MVC   SEM_NUM(2),=AL2(2)              number 2
         MVC   SEM_OP(2),=AL2(1)               release the resource
         MVC   SEM_FLG(2),=AL2(IPC_NOWAIT)     flags (nowait)
         LA    R5,SEM#BUFLEN(,R5)              ->next SEM_BUF_ELE
         MVC   SEM_NUM(2),=AL2(8)              number 8
         MVC   SEM_OP(2),=AL2(0)               test for no resource
         MVC   SEM_FLG(2),=AL2(0)              flags (wait)
         SPACE ,
         MVC   NUMB_SEM_OPS(4),=AL2(3) number of SEM_BUF_ELE in BUFFERA
         SPACE ,
         CALL  BPX1SOP,              Semaphore control operations      +
               (SEM_ID,              Input: Semaphore set ID           +
               BUFA,                 Input: ->SEM_BUF_ELE      BPXYSEM +
               NUMB_SEM_OPS,         Input: Action to take             +
               RETVAL,               Return value: 0, -1 or value      +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCESS	Permission is denied. The following reason code can accompany this return code: JRIpcDenied.
EAGAIN	The operation would result in suspension of the calling process, but the NOWAIT flag was specified. The following reason code can accompany this return code: JRIpcRetry.
EDEADLK	The combination of operations can never be satisfied. This condition is detected by analyzing the operations requested and the system maximums; it does not include interactions with other threads. For example, an operation could add 1 to a semaphore, and a later operation in the same SEM_BUF could test it for zero. The following reason code can accompany this return code: JRDeadlock.
EFAULT	The *semaphore_operations* parameter specified an address that caused the service to program check. The following reason code can accompany this return code: JRBadAddress.
EFBIG	SEM_NUM exceeds *number_of_semaphores* - 1. The following reason code can accompany this return code: JRSema4BadSemN.
EIDRM	*semaphore_set_ID* was removed from the system while the invoker was waiting. The following reason code can accompany this return code: JRIpcRemoved.

Return Code	Explanation
EINTR	The service was interrupted by a signal. The following reason code can accompany this return code: JRIpcSignaled.
EINVAL	The *semaphore_set_ID* does not represent a semaphore set. The following reason code can accompany this return code: JRIpcBadID.
ENOSPC	The space allotted for all semaphore data would be exceeded by the addition of the UNDO structure for this request. The following reason code can accompany this return code: JRSemStorageLimit.
ERANGE	An operation would cause *semval* or *semadj* to overflow the system-imposed limit. These system limits are defined in the SEM#MAX_VAL and SEM#MAX_ADJ fields of the BPXYSEM macro. The following reason codes can accompany this return code: JRSema4BadValue, JRSema4BadAdj.
E2BIG	*number_of_semaphore_operations* exceeds the maximum allowed by the system. The following reason code can accompany this return code: JRSema4BadNOps.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “cmsprocclp (BPX1MPC) — Clean Up Kernel Resources” on page 38
- “semctl (BPX1SCT) — Perform Semaphore Control Operations” on page 264
- “semget (BPX1SGT) — Create or Find a Set of Semaphores” on page 269

send (BPX1SND) — Send Data on a Socket

```
BPX1SND
        socket_descriptor
        buffer_length
        buffer
        buffer_ALET
        flags
        return_value
        return_code
        reason_code
```

Purpose

Use the send (BPX1SND) service to send data on a socket.

Parameters

socket_descriptor
(input,INT,4) is a variable for specifying the descriptor of the socket.

buffer_length
(input,INT,4) is a variable for specifying the length of the *buffer* parameter.

buffer
(input,CHAR,*buffer_length*) is a variable for specifying the data to be sent.

buffer_ALET
(input,INT,4) is a variable for specifying the access list entry token (ALET) for *buffer*.

Note: This parameter is ignored.

flags
(input,INT,4) is a variable for specifying information about how the data is to be sent. This field is mapped by the BPXYMSGF macro. See "BPXYMSGF — Map the Message Flags" on page 441.

return_value
(output,INT,4) is a variable where the service returns one of the following:

- The number of bytes sent from the buffer, if the request is successful.
- `0`, indicating the connection is closed.
- `-1`, if the request is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. The socket must be connected.

2. If there is not enough room to write the data to the output buffer, the service either blocks waiting for room, or returns an EWOULDBLOCK, depending on whether the socket is marked as blocking or nonblocking.

Example

The following code issues a send for a socket. SOCKDESC was returned previously from a call to socket (BPX1SOC). This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structures, see “BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services” on page 465 and “BPXYMSGF — Map the Message Flags” on page 441.

```
         MVC   BUFLENA,=F'16'
         MVC   BUFFERA(16),=CL16'Here is the data'
         SPACE ,
         CALL  BPX1SND,              Send data on a socket             +
               (SOCKDESC,            Input: Socket Descriptor          +
               =A(L'BUFFERA),        Input: Length of input buffer     +
               BUFFERA,              Input: Address of input buffer    +
               PRIMARYALET,          Input: Alet of input buffer       +
               MSG_FLAGS,            Input: Flags                      +
               RETVAL,               Return value: Num bytes, 0, or -1 +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	*socket_descriptor* does not refer to a valid descriptor. The following reason codes can accompany this return code: JRFileDesNotInUse, JRFileNotOpen.
ECONNRESET	Connection reset by peer.
EINTR	A signal interrupted the send before any data was written.
EIO	There has been a network or transport failure. The following reason code can accompany this return code: JRPrevSockError.
EMSGSIZE	The message is too large to be sent all at once, as the socket requires.
ENOBUFS	A buffer could not be obtained.
ENOTCONN	The socket is not connected. The following reason code can accompany this return code: JRSocketNotCon.
ENOTSOCK	*socket_descriptor* does not refer to a valid socket descriptor. The following reason code can accompany this return code: JRMustBeSocket.
EPIPE	An attempt was made to send to a socket that is shut down or closed. This error also generates a **SIGPIPE** signal.
ESHUTDOWN	There is no data to read on the socket, and it has been shut down for reading.
EWOULDBLOCK	The socket is marked nonblocking, and no space is available for data to be written.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “read (BPX1RED) — Read from a File or Socket” on page 228
- “readv (BPX1RDV) — Read Data and Store It in a Set of Buffers” on page 238
- “recv (BPX1RCV) — Receive Data on a Socket and Store It in a Buffer” on page 243
- “recvfrom (BPX1RFM) — Receive Data from a Socket and Store It in a Buffer” on page 245
- “recvmsg (BPX2RMS) — Receive Messages on a Socket and Store Them in Message Buffers” on page 248
- “sendmsg (BPX2SMS) — Send Messages on a Socket” on page 280
- “sendto (BPX1STO) — Send Data on a Socket” on page 283
- “write (BPX1WRT) — Write to a File or Socket” on page 401
- “writev (BPX1WRV) — Write Data from a Set of Buffers” on page 404

sendmsg (BPX2SMS) — Send Messages on a Socket

BPX2SMS

socket_descriptor
message_headers
flags
IOV_ALET
IOV_buffer_ALET
return_value
return_code
reason_code

Purpose

Use the sendmsg (BPX2SMS) service to send messages on a socket.

Parameters

socket_descriptor
(input,INT,4) is a variable for specifying the descriptor of the socket.

message_headers
(input,CHAR,length of BPXYMSGH) is a variable for specifying the message headers. This field is mapped by the BPXYMSGH macro. Each message header contains a pointer to an I/O vector structure, which contains information about the buffers from which the messages are to be sent. The I/O vector structure is mapped by the BPXYIOV macro. See "BPXYMSGH — Map the Message Headers" on page 443 and "BPXYIOV — Map the I/O Vector Structure" on page 430.

flags
(input,INT,4) is a variable for specifying information about how the data is to be sent. This field is mapped by the BPXYMSGF macro. See "BPXYMSGF — Map the Message Flags" on page 441.

IOV_ALET
(input,INT,4) is a variable for specifying the access list entry token (ALET) for the I/O vector (IOV) structure specified in *message_headers*.

Note: This parameter is ignored.

IOV_buffer_ALET
(input,INT,4) is a variable for specifying the ALET for the buffers that are pointed to by the IOV structure in *message_headers*.

Note: This parameter is ignored.

return_value
(output,INT,4) is a variable where the service returns the number of bytes sent from the buffers if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. The socket can be either connected or unconnected. For connected sockets, the sockaddr portion of the `msghdr` structure is ignored.
2. If there is not enough room to write the data to an output buffer, the service either blocks waiting for an output buffer to become available, or returns an EWOULDBLOCK, depending on whether the socket is marked as blocking or nonblocking.
3. When sending IPv6 Raw packets to an IPv4 mapped address, the data must be a valid IPv4 datagram, including the IPv4 header.

Example

The following code sends a message on a socket. SOCKDESC was returned from a previous call to socket (BPX1SOC). This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structures, see “BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services” on page 465, “BPXYIOV — Map the I/O Vector Structure” on page 430, and “BPXYMSGF — Map the Message Flags” on page 441.

```
         XC    MSGH,MSGH             Clear msgh
         LA    R2,SOCKADDR
         ST    R2,MSGHNAMEPTR        Store the address of sockaddr
         LA    R2,SOCK#LEN+SOCK_SUN#LEN
         ST    R2,MSGHNAMELEN
         LA    R2,IOV
         ST    R2,MSGHIOVPTR
         MVI   MSGHIOVNUM,1
*
         LA    R2,BUFFERA
         ST    R2,IOV_BASE
         LA    R2,16
         ST    R2,IOV_LEN
         MVC   BUFFERA(16),=CL16'Here is the data'
*
         CALL  BPX2SMS,              Send a message on a socket        +
               (SOCKDESC,            Input: Socket Descriptor          +
               MSGH,                 Input: Address of BPXYMSGH        +
               MSG_FLAGS,            Input: Flags                      +
               PRIMARYALET,          Input: Alet of the iov            +
               PRIMARYALET,          Input: Alet of the buffers in iov +
               RETVAL,               Return value: Num bytes or -1     +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EAFNOSUPPORT	The address family that was specified in the message header is not the same as the address family that owns the socket.
EBADF	A file descriptor that was not valid was supplied. The following reason codes can accompany this return code: JRFileDesNotInUse, JRFileNotOpen.
ECONNRESET	Connection reset by peer.
EINTR	A signal interrupted the sendmsg service before any data was written.
EINVAL	One of the input parameters was incorrect. The following reason codes can accompany this return code: JROutOfRange, JRSocketCallParmError, JRSockNoName.
EIO	There was an I/O error. The following reason code can accompany this return code: JRPrevSockError.
EMSGSIZE	The message is too large to be sent all at once, as the socket requires.

Return Code	Explanation
ENOBUFS	A buffer could not be obtained.
ENOTCONN	The socket was not connected. The following reason code can accompany this return code: JRSocketNotCon.
ENOTSOCK	*socket_descriptor* does not refer to a valid socket descriptor. The following reason code can accompany this return code: JRMustBeSocket.
EPIPE	An attempt was made to send a message to a socket that is shut down or closed. This error also generates a **SIGPIPE** signal.
ESHUTDOWN	There is no data to read on the socket, and it has been shut down for reading.
EWOULDBLOCK	The socket is marked nonblocking, and no space is available for data to be written.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Service

Another callable service related to this service is:

- “recvmsg (BPX2RMS) — Receive Messages on a Socket and Store Them in Message Buffers” on page 248

sendto (BPX1STO) — Send Data on a Socket

BPX1STO

socket_descriptor
buffer_length
buffer
buffer_ALET
flags
sockaddr_length
sockaddr
return_value
return_code
reason_code

Purpose

Use the sendto (BPX1STO) service to send data on a socket.

Parameters

socket_descriptor
(input,INT,4) is a variable for specifying the descriptor of the socket.

buffer_length
(input,INT,4) is a variable for specifying the length of the *buffer* parameter.

buffer
(input,CHAR,*buffer_length*) is a variable for specifying the buffer from which the data is to be sent.

buffer_ALET
(input,INT,4) is a variable for specifying the access list entry token (ALET) for *buffer*.

Note: This parameter is ignored.

flags
(input,INT,4) is a variable for specifying information about how the data is to be sent. This field is mapped by the BPXYMSGF macro. See "BPXYMSGF — Map the Message Flags" on page 441.

sockaddr_length
(input,INT,4) is a variable for specifying the length of the *sockaddr* parameter. This value should be less than 4096 bytes (4KB).

sockaddr
(input,INT,*sockaddr_length*) is a variable for specifying the SOCKADDR structure containing the socket address to which the data is to be sent. This field is mapped by the BPXYSOCK macro. See "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465. For connected sockets, this address is ignored.

return_value
(output,INT,4) is a variable where the service returns the number of bytes sent on the socket if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. A datagram socket can be unconnected.
2. If the sending socket has no space to hold the message that is to be transmitted, the sendto service either blocks waiting for an output buffer to become available, or returns an EWOULDBLOCK, depending on whether the socket is marked blocking or nonblocking.
3. When sending IPv6 Raw packets to an IPv4 mapped address, the data must be a valid IPv4 datagram, including the IPv4 header.

Example

The following code issues a sendto for a socket. SOCKDESC was returned from a previous call to either socket (BPX1SOC) or accept (BPX1ACP). This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structures, see “BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services” on page 465 and “BPXYMSGF — Map the Message Flags” on page 441.

```
         MVC   BUFFERA(16),=CL16'Here is the data'
         LA    R2,BUFFERA
         ST    R2,IOV_BASE
         MVI   IOV_LEN,16
         SPACE ,
         CALL  BPX1STO,              Send data to a socket             +
               (SOCKDESC,            Input: Socket Descriptor          +
               =A(L'BUFFERA),        Input: Length of the input buffer +
               BUFFERA,              Input: Address of the input buffer+
               PRIMARYALET,          Input: Alet of the input buffer   +
               MSG_FLAGS,            Input: Flags                      +
               =A(L'SOCKADDR),       Input: Length of the socket addr  +
               SOCKADDR,             Input: The socket address         +
               RETVAL,               Return value: Num bytes or -1     +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EAFNOSUPPORT	The address family that was specified in the sockaddr is not the same address family as the socket.
EBADF	A file descriptor that was not valid was specified.The following reason codes can accompany this return code: JRFileDesNotInUse, JRFileNotOpen.
ECONNRESET	Connection reset by peer. The following reason code can accompany this return code: JRSocketNotCon.
EINTR	A signal interrupted the sendto service before any data was written.
EINVAL	One of the input parameters was incorrect. The following reason codes can accompany this return code: JRSocketCallParmError, JRSockNoName.
EIO	There was an I/O error. The following reason code can accompany this return code: JRPrevSockError.
EMSGSIZE	The message is too large to be sent all at once, as the socket requires.
ENOBUFS	A buffer could not be obtained.

Return Code	Explanation
ENOTCONN	The socket was not connected. The following reason code can accompany this return code: JRSocketNotCon.
ENOTSOCK	*socket_descriptor* does not refer to a valid socket descriptor. The following reason code can accompany this return code: JRMustBeSocket.
EPIPE	An attempt was made to send to a socket that is shut down or closed. This error also generates a **SIGPIPE** signal.
EPROTOTYPE	The address specifies a socket that is not the correct type for this request.
ESHUTDOWN	There is no data to read on the socket, and it has been shut down for reading.
EWOULDBLOCK	The socket is marked nonblocking, and no space is available for data to be written.

The following are for AF_UNIX only:

Return Code	Explanation
EACCES	The process does not have search permission on a component of the path prefix, or it does not have write access to the named socket.
EIO	An I/O error occurred while reading from or writing to the file system.
ELOOP	Too many symbolic links were encountered in translating the path name in the socket address.
ENAMETOOLONG	A component of a path name exceeded NAME_MAX characters, or an entire path name exceeded PATH_MAX characters.
ENOENT	A component of the path name does not name an existing file, or the path name is an empty string.
ENOTDIR	A component of the path prefix of the path name in the socket address is not a directory.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “read (BPX1RED) — Read from a File or Socket” on page 228
- “readv (BPX1RDV) — Read Data and Store It in a Set of Buffers” on page 238
- “recv (BPX1RCV) — Receive Data on a Socket and Store It in a Buffer” on page 243
- “recvfrom (BPX1RFM) — Receive Data from a Socket and Store It in a Buffer” on page 245
- “recvmsg (BPX2RMS) — Receive Messages on a Socket and Store Them in Message Buffers” on page 248
- “send (BPX1SND) — Send Data on a Socket” on page 277
- “sendmsg (BPX2SMS) — Send Messages on a Socket” on page 280
- “write (BPX1WRT) — Write to a File or Socket” on page 401
- “writev (BPX1WRV) — Write Data from a Set of Buffers” on page 404

setegid (BPX1SEG) — Set the Effective Group ID

BPX1SEG

group_ID
return_value
return_code
reason_code

Purpose

Use the setegid (BPX1SEG) service to set the effective group ID (GID) of a process.

Parameters

group_ID
(input,INT,4) is a variable for specifying the group ID the calling process wishes to assume.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. If *group_ID* is equal to the real group ID or saved set-group-ID of the process, the effective group ID is set to *group_ID*.
2. If the calling process has the appropriate privileges, the effective group ID is set to *group_ID*. Refer to "Authorization" on page 10 for information on appropriate privileges.
3. The setegid (BPX1SEG) service does not change any supplementary group IDs of the calling process.

Example

The following code sets the effective group ID of the invoker to 1. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551.

```
         MVC   GROUPID,=XL4'00000001' Value of new effective ID
         SPACE ,
         CALL  BPX1SEG,              Set effective group ID           +
               (GROUPID,             Input: Group ID                  +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECMSERR	A CMS error was detected during CP processing. Either the POSIX communication area was not previously defined to CP, or the active PID in the POSIX communication area was not allocated to the caller. The following reason code can accompany this return code: JrInternalError.
ECPERR	An error was detected during CP processing. Either the parameter list passed to CP contained incorrect values, the specified group ID was not found in the group database, or the group database contained invalid data or was inaccessible. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JrCPInternalError.
EINVAL	The *group_ID* specified is invalid or undefined.
EPERM	The process does not have the appropriate privileges to set the group ID. Refer to "Authorization" on page 10 for information on appropriate privileges.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, "Return Codes," on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, "Reason Codes," on page 495.

Related Services

Other callable services related to this service are:

- "exec (BPX1EXC) — Run a Program" on page 72
- "getegid (BPX1GEG) — Get the Effective Group ID" on page 114
- "getgid (BPX1GID) — Get the Real Group ID" on page 116
- "setgid (BPX1SGI) — Set the Group ID" on page 290
- "setuid (BPX1SUI) — Set User IDs" on page 299.

seteuid (BPX1SEU) — Set the Effective User ID

BPX1SEU

user_ID
return_value
return_code
reason_code

Purpose

Use the seteuid (BPX1SEU) service to set the effective user ID (UID) of a process.

Parameters

user_ID
(input,INT,4) is a variable for specifying the user ID the calling process wishes to assume.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Note

If *user_ID* is the same as the process's real user ID or saved set-user-ID, or the user has the appropriate privilege, the seteuid (BPX1SEU) service sets the effective user ID to be the same as *user_ID*. See “Authorization” on page 10.

Example

The following code sets the effective user ID of the invoker to 1. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   USERID,=XL4'00000001' Value of new effective user ID
         SPACE ,
         CALL  BPX1SEU,              Set effective user ID            +
               (USERID,              Input: User ID                   +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECMSERR	A CMS error was detected during CP processing. Either the POSIX communication area was not previously defined to CP, or the active PID in the POSIX communication area was not allocated to the caller. The following reason code can accompany this return code: JrInternalError.
ECPERR	An error was detected during CP processing. Either the parameter list passed to CP contained incorrect values, the specified user ID was not found in the user database, or the user database contained invalid data or was inaccessible. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JrCPInternalError.
EINVAL	The *user_ID* specified is invalid or undefined.
EPERM	The process does not have the appropriate privileges to set the user ID. See “Authorization” on page 10.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “geteuid (BPX1GEU) — Get the Effective User ID” on page 115
- “getuid (BPX1GUI) — Get the Real User ID” on page 141
- “setuid (BPX1SUI) — Set User IDs” on page 299.

setgid (BPX1SGI) — Set the Group ID

BPX1SGI

group_ID
return_value
return_code
reason_code

Purpose

Use the setgid (BPX1SGI) service to set the real, effective, and saved-set group IDs (GIDs) for the calling process.

Parameters

group_ID
(input,INT,4) is a variable for specifying the group ID the calling process wishes to assume.

return_value
(output,INT,4) is a variable where the service returns 0 if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. If *group_ID* is equal to the real group ID or saved set-group-ID of the process, the effective group ID is set to *group_ID*.
2. If the calling process has the appropriate privileges, then the real, saved set, and effective group IDs are set to *group_ID*. See "Authorization" on page 10.
3. The setgid (BPX1SGI) service does not change any supplementary group IDs of the calling process.

Example

The following code sets the real, effective, and save group IDs to 1. The caller has an effective user UD of 0. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551.

```
         MVC   USERID,=XL4'00000001' Value of new group user ID
         SPACE ,
         CALL  BPX1SGI,             Set group ID                     +
               (GROUPID,            Input: Group ID                  +
               RETVAL,              Return value: 0 or -1            +
               RETCODE,             Return code                      +
               RSNCODE),            Reason code                      +
               VL,MF=(E,PLIST)      ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECMSERR	A CMS error was detected during CP processing. Either the POSIX communication area was not previously defined to CP, or the active PID in the POSIX communication area was not allocated to the caller. The following reason code can accompany this return code: JRInternalError.
ECPERR	An error was detected during CP processing. Either the parameter list passed to CP contained incorrect values, the specified group ID was not found in the group database, or the group database contained invalid data or was inaccessible. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRCPInternalError.
EINVAL	The *group_ID* specified is invalid or undefined.
EPERM	The process does not have the appropriate privileges to set the group ID. See “Authorization” on page 10.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “exec (BPX1EXC) — Run a Program” on page 72
- “getegid (BPX1GEG) — Get the Effective Group ID” on page 114
- “getgid (BPX1GID) — Get the Real Group ID” on page 116
- “setegid (BPX1SEG) — Set the Effective Group ID” on page 286
- “setuid (BPX1SUI) — Set User IDs” on page 299.

setopen (BPX1VM6) — Perform OpenExtensions Platform Set Functions

BPX1VM6

function_code
return_value
return_code
reason_code

Purpose

Use the setopen (BPX1VM6) service to set certain flags specific to the OpenExtensions platform without creating a new POSIX process in the virtual machine.

Parameters

function_code
(input,INT,4) is a variable for specifying the function to be performed. This variable is mapped by the BPXYVM6 macro. See "BPXYVM6 — Map the Function Code Values for the setopen Service" on page 483. The possible function codes are:

Function Code	Meaning
VM6_EXECLEVEL_OFF	Turn off the exec level processing flag so CMS will create a new POSIX process when invoking an OpenExtensions service.
VM6_EXECLEVEL_ON	Turn on the exec level processing flag so CMS will **not** create a new POSIX process when invoking an OpenExtensions service.

return_value
(output,INT,4) is a variable where the service returns `0` if the request completes successfully, or `-1` if the request is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. OpenExtensions services make use of CMS Multitasking services. An application that uses OpenExtensions services cannot issue OpenExtensions calls from interrupt handlers and cannot use non-CMS Multitasking wait services.

Example

For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structure, see "BPXYVM6 — Map the Function Code Values for the setopen Service" on page 483.

```
         LA    R15,VM6_EXECLEVEL_ON
         ST    R15,VMFUNC
         SPACE ,
         CALL  BPX1VM6,              Perform OpenExtensions set func   +
```

```
                (VMFUNC,              Input: setopen, BPXYVM6           +
                RETVAL,               Return value: 0 or -1             +
                RETCODE,              Return code                       +
                RSNCODE),             Reason code                       +
                VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EINVAL	The *function_code* parameter is incorrect.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

setpgid (BPX1SPG) — Set a Process Group ID for Job Control

BPX1SPG

process_ID
process_group_ID
return_value
return_code
reason_code

Purpose

Use the setpgid (BPX1SPG) service to place the calling process or a child process of the calling process in a process group. You identify the group by specifying a process group ID. You can assign a process to a different group, or you can start a new group with that process as its leader.

Parameters

process_ID
(input,INT,4) is a variable for specifying the ID of the process to be placed in the process group. If the ID is specified as `0`, the system uses the process ID of the calling process.

process_group_ID
(input,INT,4) is a variable for specifying the ID of the process group where *process_ID* is assigned. If the ID is specified as `0`, the system uses the process group ID indicated by the *process_ID* parameter.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. The process group ID to be assigned to the group must be within the calling process's session.
2. The process identified by the *process_ID* parameter:
   - Must be the calling process or a child process of the calling process
   - If it is a child process of the calling process, must not have successfully issued one of the `exec()` functions or must not have been created by one of the `spawn()` functions
   - Must be in the same session as the process that issued the service
   - Cannot be the session leader.
3. You cannot use the setpgid service to set the process group ID for a child process created by the spawn service, because spawn automatically invokes the exec service for the child. If you want to set a process group ID for the child process that is different from the process group ID of the parent, you must specify the process group ID for the child when you invoke spawn.

Characteristics and Restrictions

See the conditions described under *return_code*.

Example

The following code places the invoking process in its own process group (zeros indicate that the process group ID is to be set to the process ID). This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   PROCID,=A(0)          Process ID - current to leader
         MVC   GROUP,=A(0)           Group ID - current to leader
         SPACE ,
         CALL  BPX1SPG,              Set process group ID for Job Ctl  +
               (PROCID,              Input: Process to be placed in grp+
               GROUP,                Input: Target group               +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The value of *process_ID* matches the process ID of a child of the calling process, but the child either has successfully invoked one of the `exec()` functions or was created by one of the `spawn()` functions. Access to the target process was denied. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRSetpgidAfterSpawn.
EINVAL	The *process_group_ID* parameter is less than zero or has some other unsupported value. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRNoSuchPid.
EPERM	The calling process cannot change the process group ID of the specified process. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRPidEQSessLeader, JRPidDifferentSession, and JrPgidDifferentSession.
ESRCH	The specified *process_ID* is not that of the calling process nor of any of its children. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRNoSuchPid and JRNotDescendant.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “exec (BPX1EXC) — Run a Program” on page 72
- “getpgrp (BPX1GPG) — Get the Process Group ID” on page 129
- “setsid (BPX1SSI) — Create a Session and Set the Process Group ID” on page 297
- “tcsetpgrp (BPX1TSP) — Set the Foreground Process Group ID” on page 369.

setsid (BPX1SSI) — Create a Session and Set the Process Group ID

BPX1SSI

process_group_ID

return_code

reason_code

Purpose

Use the setsid (BPX1SSI) service to create a new session with the calling process as its session leader. The caller becomes the process group leader of a new process group.

Parameters

process_group_ID
(output,INT,4) is a variable where, if successful, the service returns the process group ID of the new group. The new process group ID is the same as the process ID of the caller.

If not successful in creating a new session, the service returns -1 as the *process_group_ID* value.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *process_group_ID* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *process_group_ID* is -1.

Usage Note

The calling process does not have a controlling terminal.

Characteristics and Restrictions

The calling process must not already be a process group leader.

Example

The following code creates a session and a process group (and is the leader of both). This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         CALL  BPX1SSI,             Create session, set process grp ID+
               (RETVAL,             Return value: -1 or new session ID+
               RETCODE,             Return code                        +
               RSNCODE),            Reason code                        +
               VL,MF=(E,PLIST)      ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EPERM	The caller is already a process group leader, or the caller's process ID matches the process group ID of some other process. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRCallerIsPgLeader.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “exec (BPX1EXC) — Run a Program” on page 72
- “_exit (BPX1EXI) — End a Process and Bypass the Cleanup” on page 79
- “getpid (BPX1GPI) — Get the Process ID” on page 130
- “kill (BPX1KIL) — Send a Signal to a Process” on page 146
- “setpgid (BPX1SPG) — Set a Process Group ID for Job Control” on page 294
- “sigaction (BPX1SIA) — Examine or Change a Signal Action” on page 315
- “spawn (BPX1SPN) — Spawn a Process” on page 333.

setuid (BPX1SUI) — Set User IDs

BPX1SUI

user_ID
return_value
return_code
reason_code

Purpose

Use the setuid (BPX1SUI) service to set the real, effective, and saved set user IDs for the current process.

Parameters

user_ID
(input,INT,4) is a variable for specifying the user ID the process wants to assume.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. If *user_ID* is the same as the process's real user ID or saved-set user ID, the setuid (BPX1SUI) service sets the effective user ID to be the same as *user_ID*.
2. If the calling process has appropriate privileges, then the real, effective, and saved-set user IDs are set to *user_ID*. See “Authorization” on page 10.

Example

The following code sets the effective user ID to 1. The calling process has an effective UID of 3 and a real UID of 1. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   USERID,=XL4'00000001' Value of new user ID
         MVC   USERID,..             User ID to be set from a getuid
         SPACE ,
         CALL  BPX1SUI,              Set user ID                      +
               (USERID,              Input: User ID to be set         +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECMSERR	A CMS error was detected during CP processing. Either the POSIX communication area was not previously defined to CP, or the active PID in the POSIX communication area was not a PID allocated to the caller. The following reason codes can accompany this return code: JrInternalError.
ECPERR	An error was detected during CP processing. Either the parameter list passed to CP contained incorrect values, the specified user ID was not found in the user database, or the user database contained invalid data or was inaccessible. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JrCPInternalError.
EINVAL	The user ID specified is invalid or undefined.
EPERM	The process does not have the appropriate privileges to set the user ID. See “Authorization” on page 10.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “exec (BPX1EXC) — Run a Program” on page 72
- “geteuid (BPX1GEU) — Get the Effective User ID” on page 115
- “getuid (BPX1GUI) — Get the Real User ID” on page 141
- “seteuid (BPX1SEU) — Set the Effective User ID” on page 288
- “setgid (BPX1SGI) — Set the Group ID” on page 290.

shmat (BPX1MAT) — Attach a Shared Memory Segment

BPX1MAT

shared_memory_ID
shared_memory_address
shared_memory_flag
return_value
return_code
reason_code

Purpose

Use the shmat (BPX1MAT) service to attach a shared memory segment.

Parameters

shared_memory_segment_ID
(input,INT,4) is a variable for specifying the shared memory segment identifier. This value is obtained by the shmget (BPX1MGT) service.

shared_memory_address
(input,INT,4) is a variable for specifying the address in the caller's address space where storage is to be obtained and the shared memory segment is to be attached. This must be 0, which specifies that the segment is to be attached at the first available address selected by the system on a page boundary.

shared_memory_flag
(input,INT,4) is a variable for specifying additional characteristics:

SHM_RDONLY
Specifies that the segment is to be attached for read only. Otherwise, the segment is attached for read and write.

SHM_RND
Causes the storage address specified in *shared_memory_address* to be truncated to a page boundary (that is, the last 12 bits will be zero).

These flags are defined in the BPXYSHM macro. See "BPXYSHM — Map Interprocess Communications Shared Memory Segments" on page 461.

return_value
(output,INT,4) is a variable where the service returns the address of the segment if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. If an attempt is made to access memory outside the shared memory segment, normal address space storage is accessed.

2. It is the application's responsibility to determine the length of the shared memory segment that is attached.
3. If the SHM_RDONLY flag is set, read-only access is enforced only for subsequent calls to shared memory segment (shm*xxx*) services. It cannot be enforced to prevent actual updating of the shared memory segment. It is the responsibility of the application to behave correctly.
4. Because of the nature of mapping a shared memory segment to different addresses within the multiple processes it is attached to, relative addresses should be used as pointers within the shared memory segment.

Characteristics and Restrictions

The invoker is restricted by ownership, read, and read-write permissions defined by the shmget (BPX1MGT) and shmctl (BPX1MCT) services.

Example

The following code attaches a shared memory segment. For the data structure, see “BPXYSHM — Map Interprocess Communications Shared Memory Segments” on page 461.

```
         CALL  BPX1MAT,              Shared memory segment control     +
               (SHM_ID,              Input: Shared memory segment ID   +
               SEGADDR,              Input: ST loc for seg address     +
               =A(0),                Input: Flags              BPXYSHM +
               RETVAL,               Return value: 0, -1 or value      +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	Operation permission is denied to the caller. The combination of *shared_memory_flag* and permissions denies the requester access. The following reason code can accompany this return code: JRIpcDenied.
EINVAL	One or more of the following conditions exist: • *shared_memory_segment_ID* is not a valid shared memory segment identifier. • *shared_memory_address* is not zero. • *shared_memory_address* is not on a page boundary, and SHM_RND was not specified. The following reason codes can accompany this return code: JRIpcBadID, JRBadAddress, JRNotKey8.
EMFILE	The number of shared memory segments attached to the caller's process exceeds the system-imposed maximum. The following reason code can accompany this return code: JRShmMaxAttach.
ENOMEM	The available system storage is not large enough to accommodate the shared memory segment. The following reason codes can accompany this return code: JRNoUserStorage, JRSMNoStorage, JRIarvserv, JRShrStgShortage.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “shmctl (BPX1MCT) — Perform Shared Memory Segment Control Operations” on page 304
- “shmdt (BPX1MDT) — Detach a Shared Memory Segment” on page 307
- “shmget (BPX1MGT) — Create or Find a Shared Memory Segment” on page 309

shmctl (BPX1MCT) — Perform Shared Memory Segment Control Operations

BPX1MCT

shared_memory_segment_ID
command
buffer_address
return_value
return_code
reason_code

Purpose

Use the shmctl (BPX1MCT) service to do various shared memory segment control operations, including getting status, changing variables, and removing a segment from the system.

Parameters

shared_memory_segment_ID
(input,INT,4) is a variable for specifying the shared memory segment identifier. This value is returned by the shmget (BPX1MGT) service.

command
(input,INT,4) is a variable for specifying a command that identifies the operation to be performed. The command constants are defined in the BPXYIPCP macro. See "BPXYIPCP — Map Interprocess Communications Permissions" on page 431. The possible commands are:

Command
Operation

IPC_STAT
Obtains status information about *shared_memory_segment_ID*, if the current process has read permission. This information is stored in the area pointed to by the *buffer_address* parameter and mapped by the SHMID_DS data structure in the BPXYSHM macro.

IPC_SET
Sets the values of IPC_UID, IPC_GID, and IPC_MODE for *shared_memory_segment_ID*. The values to be set are taken from the SHMID_DS data structure pointed to by the *buffer_address* parameter. You can specify any values for IPC_UID and IPC_GID. For IPC_MODE, you can specify only the mode bits defined for the *shared_memory_flags* parameter of the shmget (BPX1MGT) service.

Note: The IPC_ values set with this command are defined in the BPXYIPCP macro and mapped into the SHM_PERM field of the SHMID_DS structure in the BPXYSHM macro. In addition, the IPC_MODE field in BPXYIPCP is mapped by the BPXYMODE macro.

IPC_RMID
Removes *shared_memory_segment_ID* from the system. This operation removes the identifier and destroys the segment and the data structure associated with it.

The IPC_SET and IPC_RMID operations can be performed only by a process that has either appropriate privileges or an effective user ID equal to the value of IPC_CUID or IPC_UID in the SHMID_DS data structure associated with *shared_memory_segment_ID*.

For the SHMID_DS data structure, see "BPXYSHM — Map Interprocess Communications Shared Memory Segments" on page 461.

buffer_address
(input,INT,4) is a variable for specifying the address of the buffer to be used for shared memory segment information. The buffer is mapped by the SHMID_DS data structure in the BPX1SHM macro.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. The shmctl (BPX1MCT) service assumes that the size of the buffer pointed to by the *buffer_address* parameter is at least as large as the SHMID_DS data structure.
2. The IPC_SET operation can change permissions, which may affect the ability of a thread to use the shared memory segment callable services.
3. The IPC_MODE permissions in effect at the time a process attaches a segment will remain even if the permissions are changed by the IPC_SET operation.
4. When an IPC_RMID command is processed, no further attaches are allowed. The shared memory segment is not removed from the system until all users have called the shmdt (BPX1MDT) service to detach the segment or have terminated.
5. If an IPC_RMID command is processed before a call to the fork (BPX1FRK) service, the child is not attached to the shared memory segment.

Characteristics and Restrictions

The invoker is restricted by the ownership, read, and read-write permissions for the specified shared memory segment as defined by the shmget (BPX1MGT) and shmctl (BPX1MCT) services.

Example

The following code retrieves the size of the shared memory segment. For the data structure, see "BPXYSHM — Map Interprocess Communications Shared Memory Segments" on page 461.

```
         LA    R15,BUFFERA
         ST    R15,BUFA
         SPACE ,
         CALL  BPX1MCT,              Shared memory segment control     +
               (SHM_ID,              Input: Shared memory segment ID   +
               =A(IPC_STAT),         Input: Command            BPXYIPCP+
               BUFA,                 Input: ->SHMID_DS or 0    BPXYSHM +
               RETVAL,               Return value: 0, -1 or value      +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The IPC_STAT command was specified, but the calling process does not have read permission. The following reason code can accompany this return code: JRIpcDenied.

Return Code	Explanation
EFAULT	The *buffer_address* parameter specified an address that caused the service to program check. The following reason code can accompany this return code: JRBadAddress.
EINVAL	One of the following conditions is true: • *shared_memory_segment_ID* is not a valid shared memory segment identifier. • *command* is not a valid command. • The mode bits set by the IPC_SET command were not valid. The following reason codes can accompany this return code: JRIpcBadFlags, JRIpcBadID, JRBadEntryCode.
EPERM	The IPC_RMID or IPC_SET command was specified, but the caller has neither appropriate privileges nor an effective user ID equal to the value of IPC_CUID or IPC_UID in the SHMID_DS data structure associated with *shared_memory_segment_ID*. The following reason code can accompany this return code: JRIpcDenied.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “w_getipc (BPX1GET) — Query Interprocess Communications” on page 391
- “shmat (BPX1MAT) — Attach a Shared Memory Segment” on page 301
- “shmdt (BPX1MDT) — Detach a Shared Memory Segment” on page 307
- “shmget (BPX1MGT) — Create or Find a Shared Memory Segment” on page 309

shmdt (BPX1MDT) — Detach a Shared Memory Segment

BPX1MDT

shared_memory_address

return_value

return_code

reason_code

Purpose

Use the shmdt (BPX1MDT) service to detach a shared memory segment.

Parameters

shared_memory_address
(input,INT,4) is a variable for specifying the starting address of a shared memory segment. This is the return value from the shmat (BPX1MAT) service.

return_value
(output,INT,4) is a variable where the service returns `0` if the request was successful, or `-1` if it was unsuccessful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Characteristics and Restrictions

The invoker is restricted by ownership, read, and read-write permissions defined by the shmget (BPX1MGT) and shmctl (BPX1MCT) services.

Example

The following code detaches a shared memory segment. For the data structure, see "BPXYSHM — Map Interprocess Communications Shared Memory Segments" on page 461.

```
         CALL  BPX1MDT,              Shared memory segment detach      +
               (SEGADDR,             Input: Shared memory segment addr +
               RETVAL,               Return value: 0, -1 or value      +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EINVAL	*shared_memory_address* is not the data segment start address of a shared memory segment attached to the caller's process. The following reason code can accompany this return code: JRBadAddress.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “shmat (BPX1MAT) — Attach a Shared Memory Segment” on page 301
- “shmctl (BPX1MCT) — Perform Shared Memory Segment Control Operations” on page 304
- “shmget (BPX1MGT) — Create or Find a Shared Memory Segment” on page 309

shmget (BPX1MGT) — Create or Find a Shared Memory Segment

BPX1MGT

key
shared_memory_size
shared_memory_flags
return_value
return_code
reason_code

Purpose

Use the shmget (BPX1MGT) service to create a new shared memory segment or find an existing shared memory segment (if the user is allowed to access it). The service returns a system-assigned shared memory segment identifier.

Parameters

key
(input,INT,4) is a variable for specifying a user-defined value that identifies a shared memory segment. The *key* serves as a lookup value to determine if an associated shared memory segment identifier already exists. If an associated shared memory segment identifier does not already exist, the *key* value becomes associated with the shared memory segment identifier created by this request.

The reserved key value IPC_PRIVATE may also be specified. IPC_PRIVATE is sometimes used when a process does not want to share a memory segment or when it wants to privately control access to the memory segment by other processes. The IPC_PRIVATE constant is defined in the BPXYIPCP macro. See "BPXYIPCP — Map Interprocess Communications Permissions" on page 431.

shared_memory_size
(input,INT,4) is a variable for specifying the number of bytes of shared memory that are required.

shared_memory_flags
(input,INT,4) is a variable for specifying the type of action to be performed and the permissions to be assigned. Valid values for this parameter include any combination of the following flags (additional bits will cause an EINVAL return code):

- These flags are defined in the BPXYIPCP macro and the values are mapped onto the S_TYPE field in the BPXYMODE macro:

  Value
  Action

  IPC_CREAT
  Creates a shared memory segment if the specified *key* is not associated with a shared memory segment identifier. IPC_CREAT is ignored when the IPC_PRIVATE reserved key is specified.

  IPC_EXCL
  Causes the service to fail if the specified *key* has an associated shared memory segment identifier. IPC_EXCL is ignored when the IPC_PRIVATE reserved key is specified or the IPC_CREAT flag is not set.

- These values are defined in the BPXYMODE macro and are a subset of the access permissions that apply to files:

  S_IRUSR
  Permits the process that owns the memory segment to read it.

S_IWUSR
Permits the process that owns the memory segment to alter it.

S_IRGRP
Permits the group associated with the memory segment to read it.

S_IWGRP
Permits the group associated with the memory segment to alter it.

S_IROTH
Permits others to read the memory segment.

S_IWOTH
Permits others to alter the memory segment.

See "BPXYIPCP — Map Interprocess Communications Permissions" on page 431 and "BPXYMODE — Map Mode Constants" on page 437.

return_value
(output,INT,4) is a variable where the service returns the shared memory segment identifier associated with *key* if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. When a shared memory segment has been created, subsequent shmget (BPX1MGT) calls to find the existing shared memory segment must request a size that is less than or equal to the value specified when the shared memory segment was created.
2. As long as a thread knows the shared memory segment identifier and access is permitted, the thread can issue shmat (BPX1MAT), shmctl (BPX1MCT), or shmdt (BPX1MDT) calls for that segment, and shmget is not needed.
3. This service creates a data structure defined by SHMID_DS, if either of the following is true:
   - IPC_PRIVATE is specified in the *key* parameter.
   - The IPC_CREAT flag is set, and the specified *key* value does not already have a shared memory segment identifier associated with it.

   The SHMID_DS data structure is defined in the BPXYSHM macro, and some values are mapped into it from the BPXYIPCP macro. See "BPXYSHM — Map Interprocess Communications Shared Memory Segments" on page 461 and "BPXYIPCP — Map Interprocess Communications Permissions" on page 431.
4. Upon creation, the SHMID_DS data structure is initialized as follows:
   - IPC_CUID and IPC_UID are set to the effective user ID of the calling process.
   - IPC_CGID and IPC_GID are set to the effective group ID of the calling process.
   - The low-order 9-bits of IPC_MODE are equal to the low-order 9-bits of the *shared_memory_flags* parameter.
   - SHM_OTIME is set to 0 and SHM_CTIME is set to the current time.
   - The storage will be initialized to nulls when the segment is created.
5. The shared memory segment is removed from the system when the shmctl (BPX1MCT) service is called with the IPC_RMID command and all users have used the shmdt (BPX1MDT) service to detach the segment or have terminated.

Characteristics and Restrictions

There is a maximum number of shared memory segments allowed in the system.

The invoker is restricted by the ownership, read, and read-write permissions for the specified shared memory segment as defined by the shmget (BPX1MGT) and shmctl (BPX1MCT) services.

Example

The following code creates a private shared memory segment of 500 bytes. For the data structure, see "BPXYSEM — Map Interprocess Communications Semaphores" on page 459.

```
         MVC   KEY(4),=A(IPC_PRIVATE)  Local to this family
         MVI   S_TYPE,IPC_CREAT+IPC_EXCL   Must not already exist
         MVI   S_MODE1,0             Not used
         MVI   S_MODE2,S_IRUSR       All read and write permissions
         MVI   S_MODE3,S_IWUSR+S_IRGRP+S_IWGRP+S_IROTH+S_IWOTH
         SPACE ,
         CALL  BPX1MGT,              Create a set of semaphores        +
               (KEY,                 Input: Shared memory segment KEY  +
               =A(500),              Input: Segment size               +
               S_MODE,               Input: Creation flags     BPXYIPCP+
               RETVAL,               Return value: -1 or MessageQue ID +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
         SPACE ,
         ICM   R15,B'1111',RETVAL    Test return value
         BNP   PSEUDO                Branch on shmget failure
         ST    R15,SHM_ID            Store SHM_ID associated with key
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EINVAL	One or more of the following conditions exist: • A shared memory segment identifier does not exist for the specified *key*, and the *shared_memory_size* parameter is either zero or greater than the system-imposed maximum. • A shared memory segment identifier exists for the specified *key*, but the size of the segment associated with it is less than the *shared_memory_size* parameter, and the *shared_memory_size* parameter is not equal to 0. • The *shared_memory_flags* parameter includes bits not supported by this function. The following reason codes can accompany this return code: JRShmBadSize, JRIpcBadFlags.
EACCES	A shared memory segment identifier exists for the specified *key*, but access permission, as specified by the low-order 9-bits of the *shared_memory_flags* parameter (the S_ flags) is not granted. The following reason code can accompany this return code: JRIpcDenied.
EEXIST	A shared memory segment identifier exists for the specified *key*, and the IPC_CREAT and IPC_EXCL flags are both set. The following reason code can accompany this return code: JRIpcExists.
ENOENT	A shared memory segment identifier does not exist for the specified *key*, and the IPC_CREAT flag is not set. The following reason code can accompany this return code: JRIpcNoExists.

Return Code	Explanation
ENOMEM	A shared memory segment is to be created, but the amount of system storage would exceed the system-imposed limit. The following reason code can accompany this return code: JRShmMaxSpages.
ENOSPC	A shared memory segment is to be created, but the system-imposed limit on the maximum number of shared memory segment identifiers allocated system-wide would be exceeded. The following reason code can accompany this return code: JRIpcMaxIDs.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “shmat (BPX1MAT) — Attach a Shared Memory Segment” on page 301
- “shmctl (BPX1MCT) — Perform Shared Memory Segment Control Operations” on page 304
- “shmdt (BPX1MDT) — Detach a Shared Memory Segment” on page 307
- “w_getipc (BPX1GET) — Query Interprocess Communications” on page 391

shutdown (BPX1SHT) — Shut Down All or Part of a Duplex Socket Connection

BPX1SHT

socket_descriptor
how
return_value
return_code
reason_code

Purpose

Use the shutdown (BPX1SHT) service to shut down all or part of a duplex socket connection.

Parameters

socket_descriptor
(input,INT,4) is a variable for specifying the descriptor of the socket.

how
(input,INT,4) is a variable for specifying a value that indicates the condition of the shutdown:

SOCK#SHUTDOWNREAD
Ends communication from the socket (Read)

SOCK#SHUTDOWNWRITE
Ends communication to the socket (Write)

SOCK#SHUTDOWNBOTH
Ends communication both to and from the socket

Equates for these values are defined in the BPXYSOCK macro. See "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

- A shutdown for read means that future write operations from the other end of this socket are rejected. Any data that was already written before the shutdown occurred are available for the application that issued the shutdown to read. The data is read until a read is done that returns zero bytes, indicating that there is no more data for that socket.
- A shutdown for write means that any future writes by the application that issued the shutdown request are rejected.
- Regardless of the How option specified, reads are not rejected.

Example

The following code issues a shutdown to stop socket writes to this socket connection. SOCKDESC was returned from a previous call to socket (BPX1SOC). This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         SPACE ,
         CALL  BPX1SHT,              Shutdown communication           +
               (SOCKDESC,            Input: Socket Descriptor         +
               SOCK#SHUTDOWNWRITE,   Input: How - shutdown writes     +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	An incorrect file descriptor was supplied. The following reason codes can accompany this return code: JRFileDesNotInUse, JRFileNotOpen.
EINVAL	The *how* parameter is incorrect. It is not SOCK#SHUTDOWNREAD, SOCK#SHUTDOWNWRITE, or SOCK#SHUTDOWNBOTH. The following reason code can accompany this return code: JRBadEntryCode.
ENOBUFS	A buffer could not be obtained.
ENOTSOCK	*socket_descriptor* does not refer to a valid socket descriptor. The following reason code can accompany this return code: JRMustBeSocket.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

sigaction (BPX1SIA) — Examine or Change a Signal Action

BPX1SIA

signal
new_sa_handler_address
new_sa_mask
new_sa_flags
old_sa_handler_address
old_sa_mask
old_sa_flags
user_data
return_value
return_code
reason_code

Purpose

Use the sigaction (BPX1SIA) service to examine, change, or both examine and change the action associated with a specific signal. You can use this service in a multithreaded process to establish actions to take when the signal is received.

Note: The signal handlers, a set of additional signals to be masked, and flags specified by this service are shared by all threads within a process.

Parameters

signal
(input,INT,4) is a variable for specifying the number of the signal you want to examine, change, or both examine and change the action for.

new_sa_handler_address
(input,INT,4) is a variable for specifying either zero or the address of a fullword containing the new signal action:

- If zero, no new action is set for this signal.
- If not zero, the signal action is set using the options described below and in the BPXYSIGH macro. See "BPXYSIGH — Map Signal Constants" on page 462.

Constant	Description
SIG_DFL#	Take the default action for this signal.
SIG_IGN#	Ignore this signal.
Address	Address of the signal catcher function.

new_sa_mask
(input,CHAR,8) is a variable for specifying a 64-bit mask of the signals to be blocked during execution of the signal-catching function. The leftmost bit represents signal number 1, and the rightmost bit represents signal number 64. Bits set to 1 represent signals that are blocked.

You must always provide this parameter, even though it is not used when *new_sa_handler_address* is specified as 0.

new_sa_flags
(input,INT,4) is a variable for specifying the signal action flags.

You must always provide this parameter, even though it is not used when *new_sa_handler_address* is specified as 0.

You can set this parameter to the following constants defined in the BPXYSIGH macro:

Constant	Description
SA_FLAGS_DFT#	None of the following functions.
SA_NOCLDSTOP#	Do not generate **SIGCHLD** signals to the calling process when its children stop (used only when *signal* is set to **SIGCHLD**).
SA_OLD_STYLE#	This is provided for the C Compiler Runtime Library to implement old-style signal callable service functions.

old_sa_handler_address
(input,INT,4) is a variable for specifying either zero or the address of a fullword where the service returns the old (current) signal action. If you specify this parameter as 0, the old signal action, *old_sa_mask*, and *old_sa_flags* are not returned.

old_sa_mask
(output,CHAR,8) is a variable where the service returns the old (current) value of the 64-bit mask of signals blocked during execution of the signal-catching function. Bits set to 1 represent signals that are blocked.

You must always provide this parameter, even though a value is not returned when *old_sa_handler_address* is specified as 0.

old_sa_flags
(output,INT,4) is a variable where the service returns the old (current) signal action flags.

You must always provide this parameter, even though a value is not returned when *old_sa_handler_address* is specified as 0.

user_data
(input,CHAR,4) is a variable for specifying user-supplied data that is passed to the signal interface routine when the signal is delivered.

return_value
(output,INT,4) is a variable where the service returns 0 if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. If *new_sa_handler_address* value is set to the action SIG_DFL for a signal that cannot be caught or ignored, the sigaction (BPX1SIA) request is ignored and *return_value* is set to 0.
2. Setting a signal action to ignore for a signal that is pending causes the pending signal to be discarded.
3. Setting signal action SIG_IGN or catch for signals **SIGSTOP** or **SIGKILL** is not allowed.
4. Setting signal action SIG_IGN for **SIGCHLD** or **SIGIO** is not allowed.
5. The user data is delivered on a per singal basis for the specific signal specified on this invocation. This field must be respecified if user data is desired for the next signal.

6. The sigaction (BPX1SIA) caller's thread must be registered for signals. This occurs by calling the cmssigsetup (BPX1MSS) service or by being created with the pthread_create (BPX1PTC) service after signals are set up. If neither of these conditions exist, the sigaction (BPX1SIA) service fails with a return code of EINVAL and a reason code of JRNotSigSetup. See "cmssigsetup (BPX1MSS) — Set Up CMS Signals" on page 40.
7. Constants used for this callable service are defined in the BPXYSIGH macro. See "BPXYSIGH — Map Signal Constants" on page 462.

Characteristics and Restrictions

In a multithreaded process, the new signal action set by the sigaction (BPX1SIA) service changes the signal action for all threads in the process.

Example

The following code sets new action for SIGALRM to default processing and returns the previous action for SIGALARM. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structure, see "BPXYSIGH — Map Signal Constants" on page 462.

```
         XC    NEWMASK,NEWMASK       Don't block additional signals
         LA    R15,NCATCHER          New catcher (NCATCHER=0,1 | ->)
         ST    R15,NEWHANDL
         LA    R15,OCATCHER          Old catcher (NCATCHER=0,1 | ->)
         ST    R15,OLDHANDL
         SPACE ,
         CALL  BPX1SIA,              Examine or change signal action   +
               (=A(SIGALRM#),        Input: Signal constant   BPXYSIGH +
               NEWHANDL,             Input: 0, ->0, ->1 or ->catcher   +
               NEWMASK,              Input: 64Bit mask of signals      +
               =A(0),                Input: Action, BPXYSIGH           +
               OLDHANDL,             0, ->XL4 (return 0, 1 ->catcher)  +
               OLDMASK,              64 bit mask of signals            +
               OLDFLAGS,             Action, BPXYSIGH                  +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECMSERR	A CMS environmental or internal error has occurred. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRNotSigSetUp and JRWrongSsave.
EFAULT	The specified address for *new_sa_handler_address* or *old_sa_handler_address* was incorrect.
EINVAL	The specified *signal* value is incorrect or is an unsupported signal number, or an attempt was made to catch a signal that cannot be caught, or an attempt was made to ignore a signal that can not be ignored. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRInvalidSignal and JRInvalidSigact.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, "Return Codes," on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, "Reason Codes," on page 495.

Related Services

Other callable services related to this service are:

- “exec (BPX1EXC) — Run a Program” on page 72
- “kill (BPX1KIL) — Send a Signal to a Process” on page 146
- “cmssigsetup (BPX1MSS) — Set Up CMS Signals” on page 40
- “sigprocmask (BPX1SPM) — Examine or Change a Thread's Signal Mask” on page 321
- “sigsuspend (BPX1SSU) — Change the Signal Mask and Suspend the Thread Until a Signal Is Delivered” on page 324.

sigpending (BPX1SIP) — Examine Pending Signals

BPX1SIP

signal_pending_mask
return_value
return_code
reason_code

Purpose

Use the sigpending (BPX1SIP) service to return the union of the set of signals pending on the thread and the set of signals pending on the process. Pending signals at the process level are moved to the thread that called this service.

Parameters

signal_pending_mask
(output,CHAR,8) is a variable where the service returns a 64-bit signal pending mask. Each bit set on (set to 1) represents a signal that is 1) currently pending at either the process level or the thread level and 2) blocked by the current thread's signal mask. The leftmost bit represents signal 1, and the rightmost bit represents signal 64.

return_value
(output,INT,4) is a variable where the service returns 0 if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Characteristics and Restrictions

See Appendix E, “The Relationship of OpenExtensions Signals to Callable Services,” on page 557.

Example

The following code retrieves the mask used for pending and blocked signals. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         CALL  BPX1SIP,             Determine pending signals         +
               (SIGRET,             Signal mask return area (XL8)     +
               RETVAL,              Return value: 0 or -1             +
               RETCODE,             Return code                       +
               RSNCODE),            Reason code                       +
               VL,MF=(E,PLIST)      ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return code:

Return_code	Explanation
ECMSERR	A CMS environmental or internal error has occurred. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRWrongSsave.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Another service related to this service is:

- “sigprocmask (BPX1SPM) — Examine or Change a Thread's Signal Mask” on page 321.

sigprocmask (BPX1SPM) — Examine or Change a Thread's Signal Mask

BPX1SPM

how
new_signal_mask
old_signal_mask
return_value
return_code
reason_code

Purpose

Use the sigprocmask (BPX1SPM) service to examine, change, or both examine and change the calling thread's signal mask.

Parameters

how
(input,INT,4) is a variable for specifying a value that identifies the action to be taken on the thread's signal mask. The following constants defined in the BPXYSIGH macro define the possible actions. See "BPXYSIGH — Map Signal Constants" on page 462.

Constant	Description
SIG_BLOCK#	Add the signals in *new_signal_mask* to those to be blocked for this thread.
SIG_UNBLOCK#	Delete the signals in *new_signal_mask* from those blocked for this thread.
SIG_SETMASK#	Replace the thread's signal mask with *new_signal_mask*.

new_signal_mask
(input,INT,4) is a variable for specifying either 0 or the address of an 8-byte area that contains the 64-bit new signal mask. The new signal mask is applied to the thread's current signal mask as specified by the *how* parameter. The leftmost bit of the signal mask represents signal number 1, and the rightmost bit represents signal number 64. Mask bits set to 1 represent signals that are blocked. If this parameter is set to 0, the signal mask is not changed and the *how* parameter is ignored.

old_signal_mask
(input,INT,4) is a variable for specifying either 0 or the address of an 8-byte area where the service returns the signal mask that was in effect prior to the call, showing the signals that were blocked. The leftmost bit in the signal mask represents signal number 1, and the rightmost bit represents signal number 64. Mask bits set to 1 represent signals that were blocked. If this parameter is set to 0, no signal mask is returned.

return_value
(output,INT,4) is a variable where the service returns 0 if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. The sigprocmask (BPX1SPM) service examines, changes, or both examines and changes the signal mask for the calling thread. This mask is called the thread's signal mask. If there are any pending unblocked signals, either at the process level or at the current thread's level after changing the signal mask, at least one of the signals is delivered to the thread before the sigprocmask (BPX1SPM) service returns.
2. In a multithreaded process, the sigprocmask (BPX1SPM) service is used to control to which thread in the process a signal generated by the kill (BPX1KIL) service is delivered. For example, if two threads in a process have **SIGUSR1** signals blocked and one thread does not, the **SIGUSR1** signal generated by the kill (BPX1KIL) service from another process is delivered to the thread that does not have the signal blocked.
3. You cannot block the **SIGKILL** and the **SIGSTOP** signals. If you call the sigprocmask (BPX1SPM) service with a request that would block those signals, that part of your request is ignored and no error is indicated.
4. A request to block signals that are not supported is accepted, and a return value of zero is returned.
5. All pending unblocked signals are moved from the process level to the current thread.

Characteristics and Restrictions

See Appendix E, “The Relationship of OpenExtensions Signals to Callable Services,” on page 557.

Example

The following code changes the signal mask to block signals 1 through 16. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYSIGH — Map Signal Constants” on page 462.

```
         LA    R15,=XL8'FFFF000000000000'  Block signals 1 thru 16
         ST    R15,NEWMASKA          New mask address
         LA    R15,OLDMASK           Old signal mask
         ST    R15,OLDMASKA          Old mask address
         SPACE ,
         CALL  BPX1SPM,              Examine or change signal mask     +
               (=A(SIG_BLOCK#),      Input: How parameter BPXYSIGH     +
               NEWMASKA,             Input: 0, ->CL8                   +
               OLDMASKA,             Input: 0 | ->returned mask        +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECMSERR	A CMS environmental or internal error occurred. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRWrongSsave.
EINVAL	The value of the *how* parameter is not one of the allowed values.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “kill (BPX1KIL) — Send a Signal to a Process” on page 146
- “cmssigsetup (BPX1MSS) — Set Up CMS Signals” on page 40
- “sigaction (BPX1SIA) — Examine or Change a Signal Action” on page 315
- “sigpending (BPX1SIP) — Examine Pending Signals” on page 319
- “sigsuspend (BPX1SSU) — Change the Signal Mask and Suspend the Thread Until a Signal Is Delivered” on page 324.

sigsuspend (BPX1SSU) — Change the Signal Mask and Suspend the Thread Until a Signal Is Delivered

BPX1SSU

signal_mask
return_value
return_code
reason_code

Purpose

Use the sigsuspend (BPX1SSU) service to replace a thread's current signal mask with a new signal mask. The thread is then suspended until delivery of a signal whose action is either to process a signal-catching service or to end the thread.

Parameters

signal_mask
(input,CHAR,8) is a variable for specifying the 64-bit signal mask that is set before waiting for a signal and during the execution of any signal catcher. The leftmost bit represent signals 1 and the rightmost bit represents signal 64. Bits set to 1 represent signals that are blocked.

return_value
(output,INT,4) is a variable where the service returns a -1 if it returns to its caller.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. The caller's thread starts running again when it receives one of the signals not blocked by the mask set by this call, or a system failure occurs that sets *return_code* to some value other than EINTR.
2. The signal mask represents a set of signals that will be blocked. Blocked signals do not "wake up" the suspended service. The signals **SIGSTOP** and **SIGKILL** cannot be blocked or ignored; they are delivered to the program no matter what the signal mask specifies.
3. If the signal action is to end the thread, the sigsuspend service does not return.
4. If the signal action is to process a signal-catching service, the signal interface routine (SIR), defined by the cmssigsetup (BPX1MSS) service, is given control with the signal mask that is used during handler processing (the PpsdSaMask field of control block BPXYPPSD). The PpsdSaMask field is set to the value specified by the *Signal_mask* parameter and the current signal mask (the PpsdCurrentMask field of control block BPXYPPSD) is set to the signal mask that existed prior to the sigsuspend service.
5. All pending unblocked signals are moved from the process level to the current thread.

Characteristics and Restrictions

See Appendix E, “The Relationship of OpenExtensions Signals to Callable Services,” on page 557.

Example

The following code replaces the invoker's current mask to block signals 1 through 16 and suspend until a signal is delivered. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
MVC   WAITMASK(8),=XL8'FFFF000000000000'  Blocks 1 thru 16
SPACE ,
CALL  BPX1SSU,              Wait for a signal                +
      (WAITMASK,            Input: Wait mask, XL8            +
      RETVAL,               Return value: -1 or not returned +
      RETCODE,              Return code                      +
      RSNCODE),             Reason code                      +
      VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECMSERR	A CMS environmental or internal error occurred. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRWrongSsave.
EINTR	A signal was received and handled successfully.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “pause (BPX1PAS) — Suspend a Process Pending a Signal” on page 197
- “sigaction (BPX1SIA) — Examine or Change a Signal Action” on page 315
- “sigpending (BPX1SIP) — Examine Pending Signals” on page 319
- “sigprocmask (BPX1SPM) — Examine or Change a Thread's Signal Mask” on page 321.

sigwait (BPX1SWT) — Wait for a Signal

BPX1SWT

signal_mask
return_value
return_code
reason_code

Purpose

Use the sigwait (BPX1SWT) service to wait for an asynchronous signal. If a signal specified in the signal set is sent to the caller of this service, the value of that signal is returned to the caller and the service ends.

Parameters

signal_mask
(input,CHAR,8) is a variable for specifying a 64-bit signal mask that contains the set of signals this task is to wait on. The leftmost bit represent signal 1, and the rightmost bit represents signal 64. Bits set to 1 represent signals that are waited on.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. If any signals specified in *signal_mask* are pending when the sigwait (BPX1SWT) service is called, the value of one of those signals is returned to the caller and that signal is cleared from the set of pending signals.
2. If none of the signals specified in *signal_mask* are pending, the sigwait (BPX1SWT) service waits until a signal specified in *signal_mask* is generated. If the signal mask is zero (no bit set on), the sigwait (BPX1SWT) service waits forever (that is, until the thread is terminated).
3. If sigwait (BPX1SWT) is called for a **SIGKILL** or **SIGSTOP** signal and a **SIGKILL** or **SIGSTOP** signal arrives, the value of the signal is not returned to the caller. Rather, the **SIGKILL** or **SIGSTOP** action occurs.
4. The current sigaction associated with a signal that is returned is not performed. (See "sigaction (BPX1SIA) — Examine or Change a Signal Action" on page 315.) This action also remains unchanged by the use of the sigwait (BPX1SWT) service.
5. If multiple threads in a process issue a sigwait (BPX1SWT) call for the same signal, only one of these threads shall return from sigwait (BPX1SWT) with the signal number if the signal was directed at the process.

Example

The following code causes the caller to wait for a signal. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   WAITMASK(8),=XL8'0004000000000000'
SPACE,
         CALL  BPX1SWT,              Wait for asynchronous signal      +
               (WAITMASK,            Input: Signal mask SIGALRM        +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return code:

Return Code	Explanation
EINVAL	The *signal_mask* argument had a signal specified that represents an incorrect signal number. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRInvalidSignal.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “kill (BPX1KIL) — Send a Signal to a Process” on page 146
- “sigprocmask (BPX1SPM) — Examine or Change a Thread's Signal Mask” on page 321.

sleep (BPX1SLP) — Suspend Execution of a Process for an Interval of Time

```
BPX1SLP
        seconds
        return_value
```

Purpose

Use the sleep (BPX1SLP) service to suspend running of the calling thread (process) until either the number of seconds specified on the call has elapsed or a signal is delivered to the calling thread to either invoke a signal-catching function or end the thread.

Parameters

seconds
(input,INT,4) is a variable for specifying the number of seconds for the calling thread to sleep. Because of processor delays, the calling thread may sleep slightly longer than this specified time.

return_value
(output,INT,4) is a variable where the service returns the "remaining sleep time" value, which is the difference between *seconds* and the number of seconds that elapsed before the thread was awakened. The return value is rounded to the nearest second. (If the thread was awakened by the ending of the elapsed time specified by *seconds*, the return value is 0.) If a signal arrives and the remaining time left in the sleep is less than a half second, a value of 0 is returned.

Usage Notes

1. The suspension can actually be longer than the requested time due to the scheduling of other activity by the system.
2. An unblocked signal received during the suspension prematurely "wakes up" the thread. The appropriate signal-handling function is then invoked to handle the signal. When that signal-handling function returns, the sleep (BPX1SLP) service returns immediately even if there is "sleep time" remaining.
3. The sleep (BPX1SLP) service returns a zero if it slept for the number of seconds specified. If the time specified by the *seconds* parameter has not elapsed when the service is interrupted due to delivery of a signal, the sleep (BPX1SLP) service returns the unslept amount of time (the requested time minus the time actually slept before the signal was delivered) in seconds. Any time consumed by signal-catching functions is not reflected in the value returned by the sleep (BPX1SLP) service.
4. The following lists usage notes for a **SIGALRM** signal generated by the alarm (BPX1ALR) or kill (BPX1KIL) calls during the execution of the sleep (BPX1SLP) call:
   - If the calling thread has **SIGALRM** blocked prior to calling the sleep (BPX1SLP) service, the sleep (BPX1SLP) service does not return when **SIGALRM** is generated and the **SIGALRM** signal is left pending when sleep (BPX1SLP) returns.
   - If the calling process has **SIGALRM** ignored when the **SIGALRM** signal is generated, then the sleep (BPX1SLP) service does not return and the **SIGALRM** signal is ignored.
   - If the calling process has **SIGALRM** set to a signal-catching function, that function interrupts the sleep (BPX1SLP) service and receives control. The sleep (BPX1SLP) service returns any unslept amount of time, as it does for any other type of signal.

5. If a signal-catching function interrupts the sleep (BPX1SLP) service and either examines or changes the time a **SIGALRM** is scheduled to be generated, the action associated with the **SIGALRM** signal is the same as when the signal-catching function interrupts any other function.
6. If a signal-catching function interrupts the sleep (BPX1SLP) service and restores a previously saved environment and does not return, the action associated with the **SIGALRM** signal that was saved prior to the sleep (BPX1SLP) service is the same as when the signal-catching function interrupts any other function.
7. When the sleep (BPX1SLP) service returns, any previous alarm time that has not elapsed is restored before any signal-catcher gets control. Signal catchers can change this alarm setting. See "alarm (BPX1ALR) — Set an Alarm" on page 18.

Characteristics and Restrictions

See Appendix E, "The Relationship of OpenExtensions Signals to Callable Services," on page 557.

Example

The following code suspends running for 8 seconds or until a signal is delivered (whichever comes first). This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551.

```
         MVC   SECONDS,=F'8'         8 seconds
         SPACE ,
         CALL  BPX1SLP,              Temporarily suspend execution     +
               (SECONDS,             Input: Sleep interval in seconds  +
               RETVAL),              Return value: 0 or sleep time     +
               VL,MF=(E,PLIST)       ----------------------------------
```

VM-Related Information

Both the alarm (BPX1ALR) service and the sleep (BPX1SLP) service use CMS Timer Services. If the process invokes TimerStopAll, any outstanding timers set by the alarm or sleep service are also canceled.

If a timer set by the alarm or sleep service is canceled by TimerStopAll or expires, a SIGALRM signal is generated and a VMTIMER event is signalled. For more information on TimerStopAll and the VMTIMER event, see *z/VM: CMS Application Multitasking*.

Related Services

Other callable services related to this service are:

- "alarm (BPX1ALR) — Set an Alarm" on page 18
- "sigaction (BPX1SIA) — Examine or Change a Signal Action" on page 315
- "sigprocmask (BPX1SPM) — Examine or Change a Thread's Signal Mask" on page 321
- "sigsuspend (BPX1SSU) — Change the Signal Mask and Suspend the Thread Until a Signal Is Delivered" on page 324.

socket (BPX1SOC) — Create a Socket

BPX1SOC

domain
type
protocol
dimension
socket_vector
return_value
return_code
reason_code

Purpose

Use the socket (BPX1SOC) service to create a socket for communication. A descriptor is returned for the socket that identifies the socket in subsequent operations.

Parameters

domain
(input,INT,4) is a variable for specifying the socket domain (address family) for the socket. Values for this field are defined in the BPXYSOCK macro. See "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465.

type
(input,INT,4) is a variable for specifying the type of socket to be created. Some of the socket types are:

SOCK#_STREAM
Provides sequenced, two-way byte streams that are reliable and connection-oriented. They support out-of-band data. This type is supported in the AF_INET, AF_INET6, AF_IUCV, and AF_UNIX domains.

SOCK#_DGRAM
Provides datagrams, which are connectionless messages of a fixed maximum length whose reliability is not guaranteed. Datagrams can be corrupted, received out of order, lost, or delivered multiple times. This type is supported in only the AF_INET and AF_INET6 domains.

SOCK#_RAW
Supports AF_INET and AF_INET6. You must be a superuser to use this type.

Values for this field are defined in the BPXYSOCK macro. See "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465.

protocol
(input,INT,4) is a variable for specifying the communication protocol. Values for this field are defined in the BPXYSOCK macro. See "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465.

dimension
(input,INT,4) is a variable for specifying a value that indicates the number of sockets to be created. The only supported value is:

SOCK#DIM_SOCKET
This invokes the socket service to create a single socket.

This value is defined in the BPXYSOCK macro. See "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465.

socket_vector
(output,INT,8) is a variable where the service stores the socket descriptor. (The first four bytes contain the socket descriptor; the second four bytes are undefined.)

return_value
(output,INT,4) is a variable where the service returns 0 if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Characteristics and Restrictions

Protocols 0, 41, 43, 50, 51, 59, and 60 are not valid for AF_INET6 raw sockets.

Example

The following code creates a stream socket in the AF_UNIX domain. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services” on page 465.

```
         CALL  BPX1SOC,              Create a socket                  +
               (=A(AF_UNIX),         Input: Domain of AF_UNIX         +
               =A(SOCK#_STREAM),     Input: Type of socket stream     +
               =A(IPPROTO_IP),       Input: Default protocol          +
               =A(SOCK#DIM_SOCKET),  Input: Dimension for single      +
               SOCKET,               Output: Socket descriptor        +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	Permission is denied.
EAFNOSUPPORT	The address family that was specified with the *domain* parameter is not supported. The following reason code can accompany this return code: JRSocketCallParmError.
EAGAIN	The resource is temporarily unavailable. The following reason code can accompany this return code: JRPfsSuspend.
ECMSPFSPERM	The physical file system encountered a system error. The following reason code can accompany this return code: JRInvalidVnode.
EINVAL	The value for *dimension* is not valid. Only SOCK#DIM_SOCKET can be specified for this parameter. The following reason code can accompany this return code: JRInvalidParms.
EIO	There was an I/O error. The following reason code can accompany this return code: JRPfsDead.
ENOBUFS	A buffer could not be obtained.

Return Code	Explanation
EPROTOTYPE	The socket type is incorrect. The following reason codes can accompany this return code: JRSocketCallParmError, JRSocketTypeNotSupported.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

spawn (BPX1SPN) — Spawn a Process

BPX1SPN

pathname_length
pathname
argument_count
argument_length_list
argument_list
environment_count
environment_data_length
environment_data_list
filedesc_count
filedesc_list
inherit_area_length
inherit_area
return_value
return_code
reason_code

Purpose

Use the spawn (BPX1SPN) service to create a child process to run the specified executable file. This service combines the semantics of the fork (BPX1FRK) and exec (BPX1EXC) services.

Parameters

pathname_length
(input,INT,4) is a variable for specifying the length of the *pathname* parameter.

pathname
(input,CHAR,*pathname_length*) is a variable for specifying the path name of the file to be run. See "Understanding Byte File System (BFS) Path Name Syntax" on page 6.

The name specified in this parameter is case-sensitive (not automatically uppercased), whether the file resides in BFS or outside of BFS. For information on how the spawn service searches for the specified file, see usage note "3" on page 335.

argument_count
(input,INT,4) is a variable for specifying the number of elements in the arrays specified in the *argument_length_list* and *argument_list* parameters. If the program needs no arguments, specify `0`.

argument_length_list
(input,INT,*argument_count*) is a variable for specifying an array of 4-byte pointers, each of which is the address of a fullword containing the length of an argument to be passed to the specified program. If the program needs no arguments, specify `0`.

argument_list
(input,INT,*argument_count*) is a variable for specifying an array of 4-byte pointers, each of which is the address of a character string to be passed to the specified program as an argument. The length of each argument is specified by the corresponding element in the *argument_length_list* parameter. If the program needs no arguments, specify `0`.

environment_count
(input,INT,4) is a variable for specifying the number of elements in the arrays specified in the *environment_data_length* and *environment_data_list* parameters. If the program needs no environment data, specify `0`.

environment_data_length
(input,INT,*environment_count*) is a variable for specifying an array of 4-byte pointers, each of which is the address of a fullword containing the length of an environment variable to be passed to the specified program. If the program does not use environment variables, specify `0`.

environment_data_list
(input,INT,*environment_count*) is a variable for specifying an array of 4-byte pointers, each of which is the address of a character string to be passed to the specified program as an environment variable. The length of each environment variable is specified by the corresponding element in the *environment_data_length* parameter. If the program does not use environment variables, specify `0`.

filedesc_count
(input,INT,4) is a variable for specifying the number of file descriptors the child process shall inherit. This is also the number of elements in the array specified in the *filedesc_list* parameter. Values from 0 to OPEN_MAX are valid. If you specify `0`, all file descriptors from the parent are inherited without remapping by the child, and the *filedesc_list* parameter is ignored.

filedesc_list
(input,INT,*filedesc_count*) is a variable for specifying an array of 4-byte values, each of which indicates how one of the child's file descriptors is to be remapped from one of the caller's (parent's) file descriptors. Except for those file descriptors designated by SPAWN_FDCLOSED in the supplied array, the child's file descriptor 0 is remapped using the first value in the *filedesc_list* array, the child's file descriptor 1 is remapped using the second value in the *filedesc_list* array, and so on. For example, assume the caller supplies an array of 3 entries with the values 7, 5, and 4. This would cause the child's file descriptor 0 to be remapped to the parent's file descriptor 7, the child's file descriptor 1 to be remapped to the parent's file descriptor 5, and the child's file descriptor 2 to be remapped to the parent's file descriptor 4. The constant SPAWN_FDCLOSED is defined in the BPXYCONS macro.

inherit_area_length
(input,INT,4) is a variable for specifying the length of the inheritance structure that is to follow. If you specify `0`, the *inherit_area* parameter is ignored.

inherit_area
(input,CHAR,INHE#LENGTH) is a variable for a data area that contains the inheritance structure for the child process. See “BPXYINHE — Map the Inheritance Structure for the spawn Service” on page 426 for the details of the inheritance structure, including the definition of INHE#LENGTH.

return_value
(output,INT,4) is a variable where the service returns the process ID of the newly created child process if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. The new process (called the *child*) inherits the following attributes from the process that calls spawn (called the *parent*):
   - Session membership
   - Real user ID
   - Real group ID
   - Supplementary group IDs

- Priority
- Working directory
- Root directory
- File creation mask
- The process group ID of the parent is inherited by the child, unless the INHESETGROUP flag in the inheritance structure is set on, which indicates that the value specified in the INHEPGROUP field is to be used to determine the child's process group. If the value in INHEPGROUP is set to INHE#NEWPGROUP, the child is placed into a new process group with a process group ID set to the child's process ID. Otherwise, the child is placed into the process group represented by the value specified in INHEPGROUP.
- Signals set to be ignored in the parent are set to be ignored in the child, unless the INHESETSIGDEF flag in the inheritance structure is set on and the INHESIGDEF field specifies an overriding value.
- The signal mask is inherited from the parent, unless the INHESETSIGMASK flag in the inheritance structure is set on and the INHESIGMASK field specifies an overriding value.

2. The new child process has the following differences from the parent process:
   - The child process has a unique process ID (PID) that does not match any active process group ID.
   - The child has a different parent process ID (namely, the process ID of the process that called spawn).
   - If the *filedesc_count* parameter is specified as 0, the child has its own copy of the parent's file descriptors except for those files that are marked FCTLCLOEXEC or FCTLCLOFORK. The files marked FCTLCLOEXEC or FCTLCLOFORK are not inherited by the child. If a value greater than 0 is specified for *filedesc_count*, the parent's file descriptor's are remapped for the child as specified in the *filedesc_list* array. Those file descriptors from *filedesc_count* through OPEN_MAX in the parent are closed in the child, as are any elements in the *filedesc_list* array that are designated SPAWN_FDCLOSED. See the BPXYCONS macro for the definition of the SPAWN_FDCLOSED constant. The FCTLCLOFORK and FCTLCLOEXEC flags have no effect on inheritance when the *filedesc_list* is used to map the child's file descriptors.
   - The FCTLCLOEXEC and FCTLCLOFORK flags are not inherited from the parent's file descriptors to the child's.
   - If the INHESETTCPGRP flag is set in the inheritance structure, INHECTLTTYFD must be set to the file descriptor associated with the controlling terminal for this session. The foreground process group for this session will be set to the PGID of this child process, thus placing the child process in the foreground process group. (This is done by issuing a tcsetpgrp() syscall as part of spawn processing.)
   - If INHESETTCPGRP is not set, the foreground process group of the session remains unchanged.
   - The process and system utilization times for the child are set to zero.
   - Any file locks previously set by the parent are not inherited by the child.
   - The child process has no alarms set (similar to the results of a call to the alarm service with Wait_time specified as zero) and has no interval timers set.
   - The child has no pending signals.
   - The child gets a new process image to run the executable file, which is not a copy of the parent's.
   - Signals set to be caught are reset to their default action.
   - If the set-user-ID mode bit of the new executable file is set, the effective user ID and saved set-user-ID mode of the process are set to the group ID of the new executable file. See “BPXYMODE — Map Mode Constants” on page 437.
   - If the set-group-ID mode bit of the new executable file is set, the effective group ID and saved set-group-ID bit of the process are set to the owner user ID of the new executable file. See “BPXYMODE — Map Mode Constants” on page 437.
3. The file to be invoked must be a relocatable executable CMS module created by the GENMOD command, the BIND command, the c89 utility, or the cxx utility. The file type does not have to be MODULE. If the file is not relocatable, results are unpredictable.

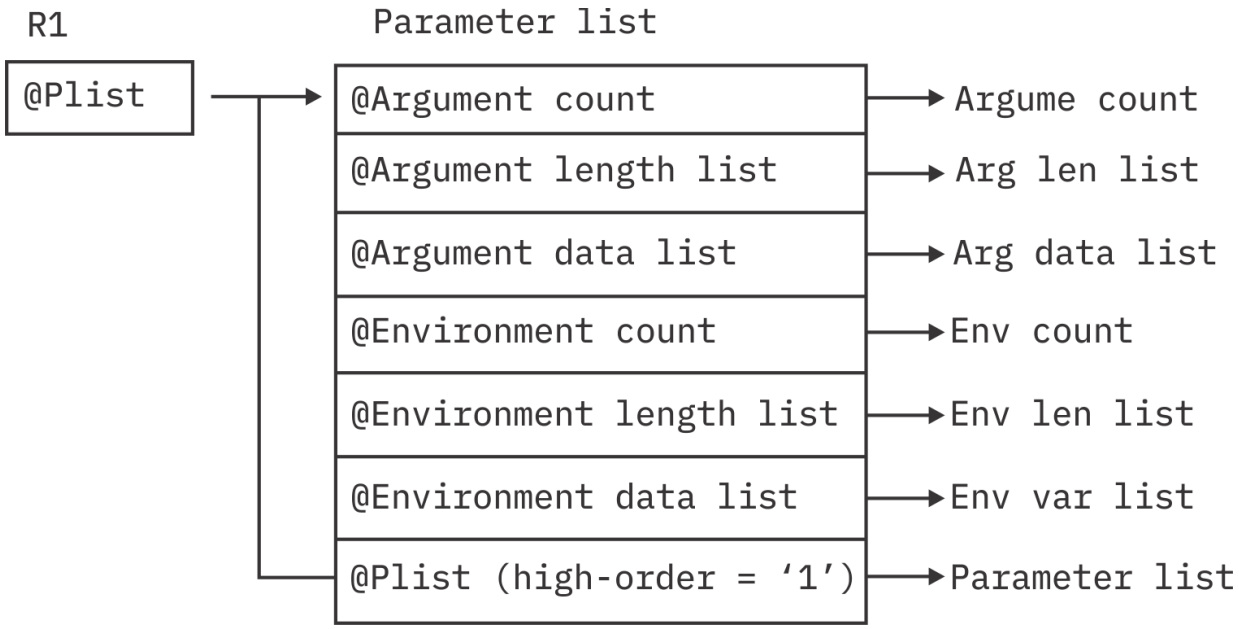
The file can reside in the byte file system or in the CMS record file system. The spawn service first looks for an executable file in the byte file system. If this fails, the service looks for an external link with a subtype of FST_EXEC. If the file is not an external link, the service parses the path name into a CMS file ID and looks for the file in the record file system.

If the file is either an external link or a CMS file ID and the file type is not specified, MODULE is assumed. If the file mode is not specified, * is assumed. If the file type is MODULE or *, and the file mode is *, the spawn service searches for a nucleus extension.

To ensure that a nucleus extension is run in the calling process, it must have been established in the CMS Commands process. Otherwise, if the nucleus extension uses OpenExtensions services, results are unpredictable.

If the file is not a nucleus extension, or no search was made for a nucleus extension because the file ID criteria described above were not met, the spawn service then searches for the file on the accessed minidisks and directories.

4. If the CMS module file to be executed contains MAP information, it is copied into the loader tables. However, because the loader tables are shared among all the processes in the virtual machine, the information in the loader tables cannot safely be relied upon in a multitasking environment.
5. The information that the service passes to the executable file to be run is a parameter list, which is pointed to by register 1. The parameter list consists of the following parameter addresses. In the last parameter address, the high-order bit is 1.



The last parameter that spawn passed to the executable file identifies the caller of the file as the exec or spawn service.

6. The child process will share the address space with its parent.
7. If the set-user-ID or set-group-ID mode bit of the executable file is set and will result in a change to the effective user ID or effective group ID, then the requestor must be authorized to have its IDs changed, and the file server on which the file resides must be authorized to change the IDs of another user.

   The following authorization applies to the requestor:

   - The External Security Manager (ESM) must grant the requestor authority to have its IDs changed, or
   - An ESM must not be installed or must defer authorization to CP, and:
     - The effective UID of the active process must be 0, or

– The requesting VM user ID must have the attribute POSIXOPT EXEC_SETIDS ALLOW set, either through a statement in its CP directory entry or through a specified or defaulted setting in the system configuration file that is not overridden in the directory entry.

The following authorization applies to the file server on which the file resides:

- The ESM must have identified to CP that the file server is authorized to change the IDs of another user when the file server logged on, or
- An ESM must not be installed or must defer authorization to CP, and the file server must have the attribute POSIXOPT SETIDS ALLOW set through a statement in its CP directory entry.

Example

The following code gives control to program ictasma located at **ict/bin** as a child process of the caller and passes arguments WK18, DEPT37A, and RATE(STD,NOEXC,NOSPEC). No environment arguments are passed. The file descriptor count is set to 0 indicating that the child shall inherit all of the parent's file descriptors. The inheritance area that is passed is set to indicate that the child process will be the process group leader of a new process group, and this process group is to be put in the foreground, with file descriptor 0 as the controlling terminal. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYINHE — Map the Inheritance Structure for the spawn Service” on page 426.

```
         MVC   BUFLENA,=F'15'
         MVC   BUFFERA(15),=C'ict/bin/ictasma'
         MVC   ARGCNT,=F'3'
*                                First
         LA    R15,4                  Length
         ST    R15,ARGLLST+00         Length parm list
         LA    R15,=CL4'WK18'         Argument
         ST    R15,ARGSLST+00         Argument address parm list
*                                Second
         LA    R15,7                  Length
         ST    R15,ARGLLST+04         Length parm list
         LA    R15,=CL7'DEPT37A'      Argument
         ST    R15,ARGSLST+04         Argument address parm list
*                                Third
         LA    R15,22                 Length
         ST    R15,ARGLLST+08         Length parm list
         LA    R15,=CL22'RATE(STD,NOEXC,NOSPEC)'    Argument
         ST    R15,ARGSLST+08         Argument address parm list
*
         MVC   ENVCNT,=F'0'           Number of env. data items passed
         MVC   ENVLENS,=F'0'          Addr of end. data length list
         MVC   ENVPARMS,=F'0'         Add of env. data
*
         MVC   FDCNT,=F'0'            Zero file descriptors passed
         MVC   FDLST,=F'0'            File descriptor list
*
         MVC   INHEEYE,=C'INHE'       Move eye catcher
         LA    R15,INHE#LENGTH        Get length of structure
         STH   R15,INHELENGTH         Put it in structure
         LA    R15,INHE#VER           Get version
         STH   R15,INHEVERSION        Put it in structure
*        Put child in new process group in foreground
         XC    INHEFLAGS,INHEFLAGS    Clear the flags
         OI    INHEFLAGS0,INHESETPGROUP+INHESETTCPGRP
         LA    R15,INHE#NEWPGROUP     Put child in new process group
         ST    R15,INHEPGROUP         Put it in structure
         LA    R15,0                  File descriptor 0
         ST    R15,INHECTLTTYFD       Controlling terminal file desc.
*
         SPACE ,
         CALL  BPX1SPN,                                              +
               (BUFLENA,              Input: Pathname length         +
               BUFFERA,               Input: Pathname                +
               ARGCNT,                Input: Argument count          +
               ARGLLST,               Input: Argument length list    +
               ARGSLST,               Input: Argument address list   +
               ENVCNT,                Input: Environment count       +
               ENVLENS,               Input: Environment length list +
               ENVPARMS,              Input: Environment address list+
               FDCNT,                 Input: File descriptor count   +
               FDLST,                 Input: File descriptor list    +
               =A(INHE#LENGTH),       Input: Length of Inheritance area +
```

```
              INHE,                Input: Inheritance area          +
              RETVAL,              Return value: -1 or not return   +
              RETCODE,             Return code                      +
              RSNCODE),            Reason code                      +
              VL,MF=(E,PLIST)      ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The caller does not have appropriate permissions to run the specified file: • The caller may lack permission to search a directory named in the *pathname* parameter. • The caller may lack execute permission for the file to be run. • The file to be run is not a regular file, and the system cannot run files of its type.
EAGAIN	The resources required to let another process be created are not available now; or you have already reached the maximum number of processes you are allowed to create. Consult the reason code to determine the exact reason the error occurred.
EBADF	An entry in the file descriptor list is not a valid file descriptor, or the controlling terminal file descriptor specified in the inheritance structure is not valid.
ECMSERR	An internal error occurred. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRNoStorage.
EINVAL	The process group ID specified in the inheritance structure is less than zero or has some other unsupported value.
ELOOP	A loop exists in symbolic links encountered during resolution of the *pathname* argument. This error is issued if more than 8 symbolic links are detected in the resolution of the path name.
EMFILE	The process has reached the maximum number of file descriptors it can have open.
ENAMETOOLONG	The *pathname* parameter is longer than 1023 characters, or some component of the path name is longer than 255 characters. CMS does not support name truncation.
ENFILE	CMS has reached the maximum number of file descriptors it can have open.
ENOENT	No path name was specified, or one or more of the components of the specified path name were not found. Consult the reason code to determine the exact reason the error occurred.
ENOEXEC	The specified file has execute permission, but is not in the proper format to be a process image file.
ENOMEM	The new process requires more memory than is permitted by the hardware or the operating system. Consult the reason code to determine the exact reason the error occurred.
ENOTDIR	A directory component of *pathname* is not a directory.

Return Code	Explanation
ENOTTY	The tcsetpgrp failed for the specified controlling terminal file descriptor in the inheritance structure. The failure occurred because the calling process does not have a controlling terminal, or the specified file descriptor is not associated with the controlling terminal, or the controlling terminal is no longer associated with the session of the calling process.
EPERM	The tcsetpgrp failed because the spawned process is not a process group leader.
ESRCH	The specified process group ID in the inheritance structure is not that of a process group in the calling process's session.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “alarm (BPX1ALR) — Set an Alarm” on page 18
- “chmod (BPX1CHM) — Change the Mode of a File or Directory by Path Name” on page 28
- “exec (BPX1EXC) — Run a Program” on page 72
- “fcntl (BPX1FCT) — Control Open File Descriptors” on page 88
- “fork (BPX1FRK) — Create a New Process” on page 96
- “sigpending (BPX1SIP) — Examine Pending Signals” on page 319
- “setpgid (BPX1SPG) — Set a Process Group ID for Job Control” on page 294
- “sigprocmask (BPX1SPM) — Examine or Change a Thread's Signal Mask” on page 321
- “stat (BPX1STA) -- Get Status Information about a File by Path Name” on page 340
- “tcsetpgrp (BPX1TSP) — Set the Foreground Process Group ID” on page 369
- “umask (BPX1UMK) — Set or Return the File Mode Creation Mask” on page 374

stat (BPX1STA) -- Get Status Information about a File by Path Name

BPX1STA

pathname_length
pathname
status_area_length
status_area
return_value
return_code
reason_code

Purpose

Use the stat (BPX1STA) service to obtain status information about a file identified by its path name. If the specified path name refers to a symbolic link, the symbolic link name is resolved to a file and the status information for that file is returned.

For the corresponding service using a file descriptor, see "fstat (BPX1FST) -- Get Status Information about a File by Descriptor" on page 102.

To obtain status information about a symbolic link, rather than for a file to which it refers, see "lstat (BPX1LST) — Get Status Information about a File or Symbolic Link by Path Name" on page 157.

Parameters

pathname_length
(input,INT,4) is a variable for specifying the length of the *pathname* parameter.

pathname
(input,CHAR,*pathname_length*) is a variable for specifying the path name of the file for which you want to obtain status. See "Understanding Byte File System (BFS) Path Name Syntax" on page 6.

status_area_length
(input,INT,4) is a variable for specifying the length of the *status_area* parameter. To determine the value of *status_area_length*, use the BPXYSTAT macro. See "BPXYSTAT — Map the File Status Structure for the stat Service" on page 473.

status_area
(input/output,CHAR,*status_area_length*) is a variable for a buffer where the the service returns the status information for the file. The status area is mapped by the BPXYSTAT macro. See "BPXYSTAT — Map the File Status Structure for the stat Service" on page 473.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. All modified data in the file identified by the *pathname* parameter is written to permanent storage when this service is requested. See "fsync (BPX1FSY) — Write Changes to Direct-Access Storage" on page 106.
2. All time fields in the *status_area* are in POSIX format, which is the number of seconds since January 1, AD 1970, 00:00:00 UTC. If you need to perform conversions on POSIX times, see the DateTimeSubtract CSL routine in the *z/VM: CMS Application Multitasking* or the DATECONVERT stage in the *z/VM: CMS Pipelines User's Guide and Reference*.
3. The File Mode field in the *status_area* is mapped by the BPXYMODE macro. See "BPXYMODE — Map Mode Constants" on page 437. For information on the values for file type, see "BPXYFTYP — Map File Type Definitions" on page 423.

Characteristics and Restrictions

To obtain information about a file, you need not have permissions for the file itself; however, you must have search permission for all the directory components of the path name.

Example

The following code obtains status about file **labrec/qual/current**. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structure, see "BPXYSTAT — Map the File Status Structure for the stat Service" on page 473.

```
         MVC   BUFFERA(19),=CL19'labrec/qual/current'
         MVC   BUFLENA,=F'19'
         SPACE ,
         CALL  BPX1STA,              Get file status                  +
               (BUFLENA,             Input: Pathname length           +
               BUFFERA,              Input: Pathname                  +
               STATL,                Input: Length of buffer needed   +
               STAT,                 Buffer, BPXYSTAT                 +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The process does not have permission to search some component of the path name prefix.
ECMSERR	An internal error occurred.
EINVAL	Parameter error—for example, a zero-length buffer. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRBuffTooSmall.
ELOOP	A loop exists in symbolic links encountered during resolution of the *pathname* argument. This error is issued if more than 8 symbolic links are detected in the resolution of the path name.
ENAMETOOLONG	The *pathname* argument is longer than 1023 characters, or some component of the path name is longer than 255 characters. This could be as a result of encountering a symbolic link during resolution of *pathname*, and the substituted string is longer than 1023 characters.

Return Code	Explanation
ENODEV	An attempt was made to use a character special file for a device not supported by OpenExtensions.
ENOENT	No file named *pathname* was found, or a path name was not specified. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRFileNotThere.
ENOTDIR	A component of the path name prefix is not a directory.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “chmod (BPX1CHM) — Change the Mode of a File or Directory by Path Name” on page 28
- “exec (BPX1EXC) — Run a Program” on page 72
- “fpathconf (BPX1FPC) — Determine Configurable Path Name Variables Using a Descriptor” on page 99
- “fstat (BPX1FST) -- Get Status Information about a File by Descriptor” on page 102
- “link (BPX1LNK) — Create a Link to a File” on page 149
- “mkdir (BPX1MKD) — Make a Directory” on page 160
- “open (BPX1OPN) — Open a File” on page 181
- “pipe (BPX1PIP) — Create an Unnamed Pipe” on page 199
- “read (BPX1RED) — Read from a File or Socket” on page 228
- “symlink (BPX1SYM) — Create a Symbolic Link to a Path Name” on page 345
- “unlink (BPX1UNL) — Remove a Directory Entry” on page 379
- “utime (BPX1UTI) -- Set File Access and Modification Times” on page 382
- “write (BPX1WRT) — Write to a File or Socket” on page 401.

statvfs (BPX1STV) — Get Status Information about a File System by Path Name

BPX1STV

pathname_length
pathname
status_area_length
status_area
return_value
return_code
reason_code

Purpose

Use the statvfs (BPX1STV) service to obtain status information about a file system identified by its path name.

For the corresponding service using a file descriptor, see "fstatvfs (BPX1FTV) — Get Status Information about File System by Descriptor" on page 104. For the corresponding service using a file system name, see "w_statvfs (BPX1STF) — Get Status Information about a File System by File System Name" on page 407.

Parameters

pathname_length
(input,INT,4) is a variable for specifying the length of the *pathname* parameter.

pathname
(input,CHAR,*pathname_length*) is a variable for specifying the path name of the file system. See "Understanding Byte File System (BFS) Path Name Syntax" on page 6 and "Understanding Network File System (NFS) Path Name Syntax" on page 9.

status_area_length
(input,INT,4) is a variable for specifying the length of the *status_area* parameter.

status_area
(output,CHAR,*status_area_length*) is a variable for the area where the service returns the status information for the file system. This area is mapped by the BPXYSSTF macro. See "BPXYSSTF — Map the File System Status Structure" on page 471.

return_value
(output,INT,4) is a variable where the service returns the length of the data returned in *status_area* if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. It is not considered an error if the passed *status_area_length* is not sufficient to hold all the returned information. (In other words, future expansion is allowed for.) As much information as will fit is written to *status_area*, and this amount is returned.
2. The amount of valid data returned in the *status_area* is indicated by the *return_value*. This allows for differences in the release levels of z/VM and the physical file systems.

Example

The following code requests information about file system containing the file identified by *pathname*.

```
MVC   BUFFERA(8),CL8'/usr/inv'
MVC   BUFLENA,=F'8'
  SPACE ,
  CALL  BPX1STV,              Get file system status           +
        (BUFLENA              Input: Pathname length           +
        BUFFERA,              Input: Pathname                  +
        SSTFL,                Input: Length of BPXYSSTF        +
        SSTF,                 Buffer, BPXYSSTF                 +
        RETVAL,               Return value: Status length or -1 +
        RETCODE,              Return code                      +
        RSNCODE),             Reason code                      +
        VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The calling process does not have permission to search some component of the path name prefix.
EAGAIN	Information is temporarily unavailable. This can occur because the mount process for the file system is incomplete.
EINVAL	Parameter error. For example, *status_area_length* is too small. The following reason code can accompany this return code: JRBuffTooSmall.
ELOOP	A loop exists in symbolic links encountered during resolution of the *pathname* argument. This error is issued if more than 24 symbolic links are detected in the resolution of path name.
ENAMETOOLONG	The *pathname* parameter is longer than 1023 characters, or a component of the path name is longer than 255 characters.
ENOENT	A component of *pathname* was not found, or no path name was specified. The following reason code can accompany this return code: JRFileNotThere.
ENOTDIR	A component of *pathname* is not a directory.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “fstatvfs (BPX1FTV) — Get Status Information about File System by Descriptor” on page 104.
- “w_statvfs (BPX1STF) — Get Status Information about a File System by File System Name” on page 407.

symlink (BPX1SYM) — Create a Symbolic Link to a Path Name

```
BPX1SYM
        pathname_length
        pathname
        link_name_length
        line_name
        return_value
        return_code
        reason_code
```

Purpose

Use the symlink (BPX1SYM) service to create a symbolic link to a path name. A file of type "symbolic link" is created.

Parameters

pathname_length
(input,INT,4) is a variable for specifying the length of the *pathname* parameter.

pathname
(input,CHAR,*pathname_length*) is a variable for specifying the path name for which you are creating a symbolic link. See “Understanding Byte File System (BFS) Path Name Syntax” on page 6.

link_name_length
(input,INT,4) is a variable for specifying the length of the *link_name* parameter.

link_name
(input,CHAR,*link_name_length*) is a variable for specifying the symbolic link being created.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

The symlink (BPX1SYM) service creates a symbolic link (*link_name*) with the file you specify (*pathname*).

Like a hard link (described in “link (BPX1LNK) — Create a Link to a File” on page 149), a symbolic link allows a file to have more than one name. The presence of a hard link guarantees the existence of a file, even after the original name has been removed. A symbolic link, however, provides no such assurance; in fact, the file identified by *pathname* need not exist when the symbolic link is created. In addition, a symbolic link can cross file system boundaries.

When a component of a path name refers to a symbolic link rather than to a directory, the path name contained in the symbolic link is resolved. If the path name in the symbolic link begins with / (slash), the symbolic link path name is resolved relative to the process root directory. If the path name in the symbolic

link does not begin with /, the symbolic link path name is resolved relative to the directory that contains the symbolic link.

If the symbolic link is not the last component of the original path name, remaining components of the original path name are resolved from there.

When a symbolic link is the last component of a path name, it may or may not be resolved. Resolution depends on the function using the path name. For example, a rename request does not have a symbolic link resolved when it appears as the final component of either the new or old path name. However, an open request does have a symbolic link resolved when it appears as the last component.

When a slash is the last component of a path name, and it is preceded by a symbolic link, the symbolic link is always resolved.

Because the mode of a symbolic link cannot be changed, its mode is ignored during the lookup process. Any files and directories to which a symbolic link refers are checked for access permission.

Example

The following code creates a symbolic link **/sysaccts** for path name **/sys12/acctn**. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   BUFFERA(12),=CL12'/sys12/acctn'
         MVC   BUFLENA,=F'12'
         MVC   BUFFERB(09),=CL09'/sysaccts'
         MVC   BUFLENB,=F'09'
         SPACE ,
         CALL  BPX1SYM,              Create symbolic link to pathname  +
               (BUFLENA,             Input: Pathname length            +
               BUFFERA,              Input: Pathname                   +
               BUFLENB,              Input: Link name length           +
               BUFFERB,              Input: Link name                  +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The requested operation requires writing in a directory with a mode that denies write permission.
EEXIST	The link name already exists. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRSymFileAlreadyExists.
EINVAL	This return code may be returned for any of the following reasons: • A component of the path prefix of the path name or the entire path name exceeds the maximum allowed. • The value of *pathname_length* is less than or equal to zero. • A null character appears in *pathname*. • The *link_name* has a slash as its last component, which indicates that the preceding component is a directory. A symbolic link cannot be a directory. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JRCompNotDir, JRInvalidSymLinkCom, JRInvalidSymLinkLen, and JRNullInPath.

Return Code	Explanation
ELOOP	A loop exists in symbolic links encountered during resolution of the *link_name* argument. This error is issued if more than 8 symbolic links are detected in the resolution of the link name.
ENAMETOOLONG	The *pathname* or *link_name* argument is longer than 1023 characters, or some component of that name is longer than 255 characters. CMS does not support name truncation.
ENOSPC	The directory in which the entry for the symbolic link is being placed cannot be extended; not enough space remains in the file system.
ENOTDIR	A component of the path prefix of *link_name* is not a directory.
EROFS	The requested operation requires writing in a directory on a read-only file system. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRReadOnlyFS.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “chown (BPX1CHO) — Change the Owner or Group of a File or Directory” on page 31
- “mkdir (BPX1MKD) — Make a Directory” on page 160
- “mknod (BPX1MKN) — Make a FIFO or Character Special File” on page 163
- “lstat (BPX1LST) — Get Status Information about a File or Symbolic Link by Path Name” on page 157
- “open (BPX1OPN) — Open a File” on page 181
- “readlink (BPX1RDL) — Read the Value of a Symbolic Link” on page 236
- “rename (BPX1REN) — Rename a File or Directory” on page 251
- “rmdir (BPX1RMD) — Remove a Directory” on page 256
- “unlink (BPX1UNL) — Remove a Directory Entry” on page 379.

sysconf (BPX1SYC) — Determine System Configuration Options

BPX1SYC

sysconf_name
return_value
return_code
reason_code

Purpose

Use the sysconf (BPX1SYC) service to get the value of a configurable system variable.

Parameters

sysconf_name
(input,INT,4) is a variable for specifying the configurable system variable to be retrieved. Each configurable system variable is mapped to a specific value as defined in the BPXYCONS macro. See "BPXYCONS — Map Constants" on page 417.

Constant	Configurable System Variable Returned
SC_ARG_MAX	The constant for ARG_MAX
SC_CHILD_MAX	The constant for CHILD_MAX
SC_CLK_TCK	The constant for CLK_TCK
SC_JOB_CONTROL	The constant for _POSIX_JOB_CONTROL
SC_NGROUPS_MAX	The constant for NGROUPS_MAX
SC_OPEN_MAX	The constant for OPEN_MAX
SC_SAVED_IDS	The constant for _POSIX_SAVED_IDS
SC_TZNAME_MAX	The constant for TZNAME_MAX
SC_VERSION	The constant for _POSIX_VERSION
SC_2_CHAR_TERM	The constant for CHAR_TERM
SC_THREAD_TASKS_MAX_NP	The constant for _THREAD_TASKS_MAX_NP

return_value
(output,INT,4) is a variable where the service returns the actual value of the configurable system variable if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Note

If the variable corresponding to *sysconf_name* exists but is not supported by the system, the sysconf service sets the return value to -1 but does not change the value of the return code.

Example

The following code gets the maximum number of children allowed by the configuration variable. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYCONS — Map Constants” on page 417.

```
         CALL  BPX1SYC,              Get configuration variable        +
               (=A(SC_CHILD_MAX),    Input: Config variable BPXYCONS   +
               RETVAL,               Return value: -1 or variable      +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return code:

Return Code	Explanation
EINVAL	The value of the *sysconf_name* argument is not valid.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Service

Another callable service related to this service is:

- “pathconf (BPX1PCF) — Determine Configurable Path Name Variables Using Path Name” on page 194.

takesocket (BPX1TAK) — Acquire a Socket from Another Program

```
BPX1TAK
        client_ID
        socket_ID
        return_value
        return_code
        reason_code
```

Purpose

Use the takesocket (BPX1TAK) service to acquire a specified socket from a specified program. A new socket descriptor is returned.

Parameters

client_ID
(input,INT,length of BPXYCID) is a variable for specifying a structure that identifies the (server) program from which the socket is to be taken. This information is typically obtained with the getclientid (BPX1GCL) service, issued by the server and passed to the taking program.

The client ID structure is mapped by the BPXYCID macro. See "BPXYCID — Map the Client ID Structure" on page 415. The structure may contain the following:

CIdDomain
Domain of the socket to be taken. Values for this field are defined in the BPXYSOCK macro. See "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465.

CIdName
The server virtual machine's user ID, left-justified and padded with blanks.

CIdTask
The server program's subtask name.

CIdReserved
Binary zeros.

socket_ID
(input,INT,4) is a variable for specifying an identifier for the socket being taken. This is supplied by the server program. It is the socket descriptor obtained from an accept (BPX1ACP) call.

return_value
(output,INT,4) is a variable where the service returns the new socket descriptor if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. The client ID output of getclientid (BPX1GCL) that is issued by the server program and passed to the secondary is intended to be used as the input client ID of the takesocket service. This identifies the program from which the socket is to be taken.

Example

The following code takes a socket that was given by the program identified by CID (client ID). SOCKDESC and CID information are passed by the program that did the givesocket (BPX1GIV). SOCKDESC is the giver's descriptor. When takesocket completes successfully, RETVAL will contain the taker's new socket descriptor. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYCID — Map the Client ID Structure” on page 415.

```
         CALL  BPX1TAK,             Take a socket from another program+
               (CID,                Input: Clientid of giver          +
               SOCKDESC,            Input: Giver's socket descriptor  +
               RETVAL,              Return value: -1 or new descriptor+
               RETCODE,             Return code                       +
               RSNCODE),            Reason code                       +
               VL,MF=(E,PLIST)      ----------------------------------
         L  R2,RETVAL
         ST R2,SOCKDES2             Store the new socket descriptor
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	Permission is denied. The following reason code can accompany this return code: JRUserNotAuthorized.
EAFNOSUPPORT	The address family is not supported.
EBADF	*socket_ID* does not specify a valid socket that is owned by the other application, or the socket has already been taken.
ECMSSTORAGE	There was a storage management error. The following reason code can accompany this return code: JRStorageReleaseErr.
EINVAL	The *client_ID* parameter does not specify a valid client identifier: either the client's process cannot be found, or the client's process was found, but it has no outstanding givesockets. The following reason code can accompany this return code: JRSocketCallParmError.
EMFILE	The socket descriptor table is already full.
EPFNOSUPPORT	The domain field of the *client_ID* parameter is not AF_INET or AF_INET6.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “accept (BPX1ACP) — Accept a Connection Request from a Client Socket” on page 12
- “getclientid (BPX1GCL) — Obtain the Calling Program's Identifier” on page 110
- “givesocket (BPX1GIV) — Give a Socket to Another Program” on page 142

tcdrain (BPX1TDR) — Wait Until Output Has Been Transmitted

BPX1TDR

file_descriptor
return_value
return_code
reason_code

Purpose

Use the tcdrain (BPX1TDR) service to wait until all output sent to a device has actually been sent.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the file descriptor that represents the output device.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. Data is considered written when it is transmitted to the terminal from the output queue.
2. The following table defines the processing of the **SIGTTOU** signal when the tcdrain (BPX1TDR) service is called from a background session against a controlling terminal:

SIGTTOU Processing	Expected Behavior
Default or signal handler	The **SIGTTOU** signal is generated. The function is not performed. The *return_value* is set to `-1`, and the *return_code* is set to EINTR.
Ignored or blocked	The **SIGTTOU** signal is not sent. The function continues normally.

Example

The following code waits until all output sent to the standard output file has been transmitted. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         CALL  BPX1TDR,              Wait for output transmittal       +
               (=A(STDOUT_FILENO),   Input: File descriptor            +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
```

```
                RSNCODE),             Reason code                        +
                VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	The *file_descriptor* argument does not describe a valid open file.
EINTR	A signal interrupted the service before all output had been sent.
ENOTTY	The specified file descriptor is not associated with a terminal.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “tcflow (BPX1TFW) — Suspend or Resume Data Flow on a Terminal” on page 354
- “tcflush (BPX1TFH) — Flush Input or Output on a Terminal” on page 356
- “tcsendbreak (BPX1TSB) — Send a Break Condition to a Terminal” on page 363.

tcflow (BPX1TFW) — Suspend or Resume Data Flow on a Terminal

```
BPX1TFW
        file_descriptor
        action
        return_value
        return_code
        reason_code
```

Purpose

Use the tcflow (BPX1TFW) service to suspend or resume data flow on a terminal.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the file descriptor for the terminal device.

action
(input,INT,4) is a variable for specifying an indicator of the action to be taken. The possible constants are mapped in the BPXYTIOS macro. See "BPXYTIOS — Map the termios Structure" on page 477.

Constant	Description
TCIOFF	Send a STOP character to the terminal to stop the terminal from sending any further input.
TCION	Send a START character to the terminal to start the terminal sending input.
TCOOFF	Suspend output to the terminal.
TCOON	Resume output to the terminal.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Note

The following table defines the processing of the **SIGTTOU** signal when the tcflow (BPX1TFW) service is called from a background session against a controlling terminal:

SIGTTOU Processing	Expected Behavior
Default or signal handler	The **SIGTTOU** signal is generated. The function is not performed. The *return_value* is set to `-1`, and the *return_code* is set to EINTR.

SIGTTOU Processing	Expected Behavior
Ignored or blocked	The **SIGTTOU** signal is not sent. The function continues normally.

Example

The following code resumes data flow (TCION transmits a START character) on the standard input file. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYTIOS — Map the termios Structure” on page 477.

```
         CALL  BPX1TFW,              Suspend or resume data flow       +
               (=A(STDIN_FILENO),    Input: File descriptor            +
               =A(TCION),            Input: Action BPXYTIOS            +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	The *file_descriptor* argument does not describe a valid open file.
EINTR	A signal interrupted the call.
EINVAL	The *action* parameter does not contain one of the expected values.
ENOTTY	The specified file descriptor is not associated with a terminal.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “tcdrain (BPX1TDR) — Wait Until Output Has Been Transmitted” on page 352
- “tcflush (BPX1TFH) — Flush Input or Output on a Terminal” on page 356
- “tcsendbreak (BPX1TSB) — Send a Break Condition to a Terminal” on page 363.

tcflush (BPX1TFH) — Flush Input or Output on a Terminal

```
BPX1TFH
        file_descriptor
        queue_selector
        return_value
        return_code
        reason_code
```

Purpose

Use the tcflush (BPX1TFH) service to flush all data sent to a device. Depending on the value of the *queue_selector* parameter, any data written but not sent, or any data received but not read, is discarded.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the file descriptor of the terminal.

queue_selector
(input,INT,4) is a variable for specifying the queues to be flushed. The constants are mapped in the BPXYTIOS macro. See "BPXYTIOS — Map the termios Structure" on page 477.

Constant	Description
TCIFLUSH	Flush data received but not read
TCOFLUSH	Flush data written but not sent
TCIOFLUSH	Flush both data received but not read and data written but not sent

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Note

The following table defines the processing of the **SIGTTOU** signal when the tcflush (BPX1TFH) service is called from a background session against a controlling terminal:

SIGTTOU Processing	Expected Behavior
Default or signal handler	The **SIGTTOU** signal is generated. The function is not performed. The *return_value* is set to `-1`, and the *return_code* is set to EINTR.

SIGTTOU Processing	Expected Behavior
Ignored or blocked	The **SIGTTOU** signal is not sent. The function continues normally.

Example

The following code flushes all the data in the standard input file. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYTIOS — Map the termios Structure” on page 477.

```
         CALL  BPX1TFH,              Line control flush               +
               (=A(STDIN_FILENO),    Input: File descriptor           +
               =A(TCIFLUSH),         Input: Queue selector BPXYTIOS   +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	The *file_descriptor* argument is not a valid open file descriptor.
EINTR	A signal interrupted the call.
EINVAL	The *queue_selector* specified was incorrect.
ENOTTY	The file associated with the file descriptor is not a terminal.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “tcdrain (BPX1TDR) — Wait Until Output Has Been Transmitted” on page 352
- “tcflow (BPX1TFW) — Suspend or Resume Data Flow on a Terminal” on page 354
- “tcsendbreak (BPX1TSB) — Send a Break Condition to a Terminal” on page 363.

tcgetattr (BPX1TGA) — Get the Attributes for a Terminal

BPX1TGA

file_descriptor
termios_structure
return_value
return_code
reason_code

Purpose

Use the tcgetattr (BPX1TGA) service to get control information for a terminal and store it in a termios data area that you provide.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the file descriptor of the terminal for which you want attributes.

termios_structure
(output,CHAR,BPXYTIOS#LENGTH field in BPXYTIOS macro) is a variable for an area where the service returns a structure that contains the terminal control modes, input modes, output modes, local modes, and special control characters as defined by the POSIX standard. This structure is mapped by the BPXYTIOS macro. See "BPXYTIOS — Map the termios Structure" on page 477.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. The BPXYTIOS macro should be used to map the termios structure and define the equates for bits and values. Note the following about BPXYTIOS:
   - BPXYTIOS generates standard POSIX-defined names, with the exception that all names are uppercase. In addition, all names can have a user-specified prefix.
   - When testing or setting bits in flag fields, you should use an offset name to define which byte in the flag field contains the bit. For instance: TM C_CFLAG+HUPCL_O,HUPCL.
   - CS5 through CS8 values can be contained in CSIZE. CSIZE is essentially a 2-bit integer that can contain decimal values `0` through 3, as defined by CS5 through CS8.
   - BPXYTIOS can be used to define either a DSECT or an inline structure. This is determined by the DSECT= keyword.
   - The C_CC field is an array of 1-byte fields, indexed by the various special character equates. These equates can be used as offsets into C_CC, or can be put into a register to be used with indexing instructions. For instance:

```
MVC   C_CC+VSUSP,NEWVAL    To set a new value
LA    R10,VSUSP            To set an register to use as an index
                              in a later IC or STC instructions
```

2. You can run the tcgetattr (BPX1TGA) service either in a foreground or in a background process. However, if the process is in the background, a foreground process can later change the attributes that you obtained.

Example

The following code retrieves control information about the standard input file. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYTIOS — Map the termios Structure” on page 477.

```
         CALL  BPX1TGA,              Get a terminal control structure  +
               (=A(STDIN_FILENO),    Input: File descriptor            +
               TIOS,                 Termio structure, BPXYTIOS        +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	The *file_descriptor* argument is not a valid open file descriptor.
ENOTTY	The file associated with the file descriptor is not a terminal, or the process does not have a controlling terminal, or the file is not the controlling terminal for the process.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Service

Another callable service related to this service is:

- “tcsetattr (BPX1TSA) — Set the Attributes for a Terminal” on page 365.

tcgetpfx (BPX1TGX) — Get the Control Sequence Prefix

BPX1TGX

control_character_prefix

Purpose

Use the tcgetpfx (BPX1TGX) service to obtain the control sequence prefix for the terminal associated with the calling process.

Parameters

control_sequence_prefix
(output,CHAR,1) is a variable where the service returns the control sequence prefix.

Example

The following code retrieves the prefix for the terminal. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         CALL  BPX1TGX,              Determine prefix                 +
               (PREFIX),             Output: Prefix                   +
               VL,MF=(E,PLIST)       ---------------------------------
```

Related Service

Another callable service related to this service is:

- “tcsetpfx (BPX1TSX) — Set the Control Sequence Prefix” on page 368

tcgetpgrp (BPX1TGP) — Get the Foreground Process Group ID

BPX1TGP

file_descriptor
return_value
return_code
reason_code

Purpose

Use the tcgetpgrp (BPX1TGP) service to get the process group ID of the foreground process group associated with a terminal identified by its file descriptor.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the file descriptor for the terminal.

return_value
(output,INT,4) is a variable where the service returns the process group ID of the foreground process group associated with the terminal if the request is successful, or -1 if it is not successful. If there is no foreground process group, a positive value, not equal to any existing process group, is returned.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Example

The following code gets the foreground process group ID associated with the controlling terminal. For this example to work, STDIN must be associated with the controlling terminal. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         CALL  BPX1TGP,              Get the foreground process grp ID +
               (=A(STDIN_FILENO),    Input: File descriptor            +
               RETVAL,               Return value -1, fgrd proc grp ID +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	The *file_descriptor* argument does not specify a valid open file descriptor.
ENOTTY	The file descriptor is not associated with a terminal.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “setpgid (BPX1SPG) — Set a Process Group ID for Job Control” on page 294
- “setsid (BPX1SSI) — Create a Session and Set the Process Group ID” on page 297
- “tcsetpgrp (BPX1TSP) — Set the Foreground Process Group ID” on page 369.

tcsendbreak (BPX1TSB) — Send a Break Condition to a Terminal

```
BPX1TSB
        file_descriptor
        duration
        return_value
        return_code
        reason_code
```

Purpose

Use the tcsendbreak (BPX1TSB) service to send a BREAK signal to a terminal that uses asynchronous serial data transmission.

Note: Because OpenExtensions terminals do not use asynchronous serial data transmission, this function does not send a BREAK signal. Instead, it returns without any action.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the file descriptor for the terminal device to which the break is to be sent.

duration
(input,INT,4) is a variable for specifying the duration of the break transmission.

Note: Because this service has no effect on OpenExtensions terminals, the *duration* value is ignored.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. The following table defines the processing of the SIGTTOU signal when the tcsendbreak (BPX1TSB) service is called from a background session against a controlling terminal:

SIGTTOU Processing	Expected Behavior
Default or signal handler	The SIGTTOU signal is generated. The function is not performed. The return value is set to `-1`, and the return code is set to EINTR.
Ignored or blocked	The SIGTTOU signal is not sent. The function continues normally.

Example

The following code requests sending a break to the standard input file. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         CALL  BPX1TSB,              Send break condition to terminal  +
               (=A(STDIN_FILENO),    Input: File descriptor            +
               =A(0),                Duration, not used in OpenExtensions+
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	The *file_descriptor* argument is not a valid open file descriptor.
EINTR	The service was called from a background job, and the SIGTTOU signal had either a default action or a signal handler. The function was not performed.
ENOTTY	The specified file descriptor is not associated with a terminal.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “tcdrain (BPX1TDR) — Wait Until Output Has Been Transmitted” on page 352
- “tcflow (BPX1TFW) — Suspend or Resume Data Flow on a Terminal” on page 354
- “tcflush (BPX1TFH) — Flush Input or Output on a Terminal” on page 356.

tcsetattr (BPX1TSA) — Set the Attributes for a Terminal

BPX1STA

file_descriptor
actions
termios_structure
return_value
return_code
reason_code

Purpose

Use the tcgetattr (BPX1TSA) service to set control information for a terminal from a termios data area that you provide.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the file descriptor of the terminal for which you want to set attributes.

actions
(output,INT,4) is a variable where the service returns a value that indicates how the attributes are to be set. The following possible values for this parameter are defined in the BPXYTIOS macro. See "BPXYTIOS — Map the termios Structure" on page 477.

Constant	Description
TCSANOW	Change the terminal attributes immediately.
TCSADRAIN	Change the terminal attributes when all output to the terminal has been sent.
TCSAFLUSH	Change the terminal attributes when all output to the terminal has been sent, and all input that has been received but not read is to be discarded.

termios_structure
(input,CHAR,BPXYTIOS#LENGTH field in BPXYTIOS macro) is a variable for an area containing a termios structure in which you specify the attributes you want to set. The termios structure contains the terminal control modes, input modes, output modes, local modes, and special control characters as defined by the POSIX standard. The structure is mapped by the BPXYTIOS macro. See "BPXYTIOS — Map the termios Structure" on page 477.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. A program should always issue the tcsetattr (BPX1TSA) callable service using a termios structure returned from a previous call to the tcgetattr (BPX1TGA) service, with appropriate changes to the various fields.
2. The BPXYTIOS macro should be used to map the termios structure and define the equates for bits and values. Note the following about BPXYTIOS:
   - BPXYTIOS generates standard POSIX-defined names, with the exception that all names are uppercase. In addition, all names can have a user-specified prefix.
   - When testing or setting bits in flag fields, you should use an offset name to define which byte in the flag field contains the bit. For instance: TM C_CFLAG+HUPCL_O,HUPCL.
   - CS5 through CS8 values can be contained in CSIZE. CSIZE is essentially a 2-bit integer that can contain decimal values 0 through 3, as defined by CS5 through CS8.
   - BPXYTIOS can be used to define either a DSECT or an inline structure. This is determined by the DSECT= keyword.
   - The C_CC field is an array of 1-byte fields, indexed by the various special character equates. These equates can be used as offsets into C_CC, or can be put into a register to be used with indexing instructions. For instance:

```
MVC   C_CC+VSUSP,NEWVAL    To set a new value
LA    R10,VSUSP            To set an register to use as an index
                             in a later IC or STC instructions
```

3. The following table defines the processing of the SIGTTOU signal when the tcsetattr (BPX1TSA) service is called from a background session against a controlling terminal:

SIGTTOU Processing	Expected Behavior
Default or signal handler	The SIGTTOU signal is generated. The function is not performed. The return value is set to -1, and the return code is set to EINTR.
Ignored or blocked	The SIGTTOU signal is not sent. The function continues normally.

Example

The following code turns off the HUPCL (hang up on last close) bit for the standard input file. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYTIOS — Map the termios Structure” on page 477.

```
         NI    C_CFLAG+HUPCL_O,X'FF'-HUPCL   Turn off HUPCL
*        termios was retrieved by a prior tcgetattr
         CALL  BPX1TSA,              Set terminal attributes           +
               (=A(STDIN_FILENO),    Input: File descriptor            +
               =A(TCSADRAIN),        Input: Action            BPXYTIOS +
               TIOS,                 Input: Terminos struct   BPXYTIOS +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	The *file_descriptor* argument is an incorrect open file descriptor.
EINTR	A signal interrupted the call.
EINVAL	An action or value specified was incorrect.

Return Code	Explanation
ENOTTY	The file associated with the file descriptor is not a terminal.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Service

Another callable service related to this service is:

- “tcgetattr (BPX1TGA) — Get the Attributes for a Terminal” on page 358.

tcsetpfx (BPX1TSX) — Set the Control Sequence Prefix

BPX1TSX

control_character_prefix

Purpose

Use the tcsetpfx (BPX1TSX) service to set the control sequence prefix for the terminal associated with the calling process.

Parameters

control_sequence_prefix
(input,CHAR,1) is a variable for specifying the new control sequence prefix.

Example

The following code sets the prefix for the terminal. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   PREFIX,C'!'           Put the desired prefix into var.
         SPACE ,
         CALL  BPX1TSX,              Set prefix                       +
               (PREFIX),             Input:  Prefix                   +
               VL,MF=(E,PLIST)       ---------------------------------
```

Related Service

Another callable service related to this service is:

- “tcgetpfx (BPX1TGX) — Get the Control Sequence Prefix” on page 360

tcsetpgrp (BPX1TSP) — Set the Foreground Process Group ID

BPX1TSP

file_descriptor
process_group_id
return_value
return_code
reason_code

Purpose

Use the tcsetpgrp (BPX1TSP) service to move the requested process group into the foreground, replacing the current foreground process group. The current foreground process group then becomes the background process group.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the file descriptor of the terminal device.

process_group_ID
(input,INT,4) is a variable for specifying the ID of the process group you want to have associated with the controlling terminal.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. The terminal identified by the *file_descriptor* parameter must be the controlling terminal of the calling process, and must be currently associated with the session of the calling process. The file descriptor can be any of the descriptors representing the controlling terminal (such as standard input [stdin], standard output [stdout], and standard error [stderr]). This service affects future access from any file descriptor in use for the terminal.

   Note: You must consider redirection when choosing the file descriptor to specify.
2. The *process_group_ID* must represent a process group in the same session as the calling process.
3. After the foreground process group is set, reads by the process group formerly in the foreground fail or cause the process group to stop from a SIGTTIN signal. Writes can also cause the process to stop (from a SIGTTOU signal) or can succeed, depending upon the current setting of TOSTOP (set by the tcsetattr (BPX1TSA) service) and the signal options for SIGTTOU.

Example

The following code sets the controlling terminal's foreground process group to a new value. For this example to work, STDIN must be associated with the controlling terminal. This example follows the rules

of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   PROCID,..             Process group ID set by setpgrp
         SPACE ,
         CALL  BPX1TSP,              Set foreground process group ID   +
               (=A(STDIN_FILENO),    Input: File descriptor            +
               PROCID,               Input: Foreground process group ID+
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	The *file_descriptor* argument is not a valid open file descriptor.
EINVAL	The *process_group_ID* argument is not a process group ID supported by this implementation.
ENOTTY	The calling process does not have a controlling terminal, or *file_descriptor* is not associated with the controlling terminal, or the controlling terminal is no longer associated with the session of the calling process.
EPERM	The *process_group_ID* argument does not match the process group ID of any process in the same session as the calling process.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “setpgid (BPX1SPG) — Set a Process Group ID for Job Control” on page 294
- “setsid (BPX1SSI) — Create a Session and Set the Process Group ID” on page 297
- “tcgetpgrp (BPX1TGP) — Get the Foreground Process Group ID” on page 361.

times (BPX1TIM) — Get Process and Child Process Times

BPX1TIM

time_data
return_value
return_code
reason_code

Purpose

Use the times (BPX1TIM) service to gather information about processor time used by the current process or related processes.

Parameters

time_data
(output,CHAR,16) is a variable for an area where the service returns information about processor time used. This area is mapped by the BPXYTIMS macro. See "BPXYTIMS — Map the Processor Time Structure for the times Service" on page 475.

return_value
(output,INT,4) is a variable where the service returns the number of clock ticks (hundredths of a second) that have elapsed since the current address space became a POSIX process. If this value cannot be determined, the service returns -1.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Note

Processor times for a child process that has ended are not added to the TIMSCUTIME and TIMSCSTIME of the parent process until the parent issues a wait or waitpid for that child process. See "wait (BPX1WAT) — Wait for a Child Process to End" on page 385 for more information on this subject.

Example

The following code gathers selected times about the invoker's CPU utilization. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structure, see "BPXYTIMS — Map the Processor Time Structure for the times Service" on page 475.

```
         CALL  BPX1TIM,              Process CPU times                 +
               (TIMS,                Input: Buffer            BPXYTIMS +
               RETVAL,               Return value: -1 or clock_t       +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

VM-Related Information

The TIMSSTIME value returned by the times (BPX1TIM) service is the portion of time spent in the CMS root process and is accumulated from the most recent time the CMS session became a POSIX process.

The TIMSUTIME value is the portion of time spent in the CMS user process and is accumulated from the most recent time the CMS session became a POSIX process.

Return Codes and Reason Codes

This service can return the following return code:

Return Code	Explanation
ERANGE	An overflow occurred computing time values.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “exec (BPX1EXC) — Run a Program” on page 72
- “cmsprocclp (BPX1MPC) — Clean Up Kernel Resources” on page 38
- “spawn (BPX1SPN) — Spawn a Process” on page 333
- “wait (BPX1WAT) — Wait for a Child Process to End” on page 385.

ttyname (BPX1TYN) — Get the Name of a Terminal

BPX1TYN

file_descriptor

terminal_name_length

terminal_name

Purpose

Use the ttyname (BPX1TYN) service to obtain the path name of the terminal associated with the file descriptor.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the file descriptor of the terminal.

terminal_name_length
(input,INT,4) is a variable for specifying the size in bytes of the buffer referred to by the *terminal_name* parameter.

This length should be 1024 bytes (PATH_MAX+1), unless you know the path name is shorter.

terminal_name
(output,CHAR,*terminal_name_length*) is a variable for a buffer where the service returns either of the following:

- The path name of the terminal, terminated by a X'00'
- A single byte of X'00' (null string) if the file descriptor is not valid or does not represent a terminal.

Usage Notes

1. This service does not return -1 to indicate a failure (there is no return value parameter). If the file descriptor is incorrect, a null string is returned.
2. If the *terminal_name* buffer is smaller than the actual path name of the terminal, the name is truncated.

Example

The following code retrieves the path name for the standard error output file. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   BUFLENA,=A(1023)      Maximum pathname
         CALL  BPX1TYN,              Determine terminal name           +
               (=A(STDERR_FILENO),   Input: File descriptor            +
               BUFLENA,              Length of buffer for pathname     +
               BUFFERA),             Buffer for pathname of terminal   +
               VL,MF=(E,PLIST)       ----------------------------------
```

Related Service

Another callable service related to this service is:

- “isatty (BPX1ITY) — Determine If a File Descriptor Represents a Terminal” on page 145

umask (BPX1UMK) — Set or Return the File Mode Creation Mask

BPX1UMK

file_mode_creation_mask

return_value

Purpose

Use the umask (BPX1UMK) service to change the file mode creation mask of your process. The file mode creation mask is used to turn off permission bits in the mode parameter specified. Bit positions that are set in the file mode creation mask are cleared in the mode of the created file.

Parameters

file_mode_creation_mask
(input,INT,4) is a variable for specifying the file mode creation mask. This mask turns off permission bits in the mode of files created by the process. The mask is mapped by the BPXYMODE macro. See "BPXYMODE — Map Mode Constants" on page 437.

return_value
(output,INT,4) is a variable where the service returns the previous value of the file mode creation mask. This fullword has the same mapping as the *file_mode_creation_mask* parameter.

Usage Notes

1. File permission bits turned ON in the file creation mask are turned OFF in the mode of files created by the process. For example, if a call to the open (BPX1OPN) service specifies a *mode* argument with file permission bits, any of those bits that have been set on in the file creation mask are turned off in the *mode* argument, and therefore in the mode of the created file.
2. Only the file permission bits of the new mask are used. For example, the type of file field in *file_mode* cannot be masked.

Example

The following code changes the process's file mode creation mask (to user read, group execute, other execute). This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structure, see "BPXYMODE — Map Mode Constants" on page 437.

```
         XC    S_MODE,S_MODE
         MVI   S_MODE3,S_IXUSR+S_IXGRP+S_IXOTH  Search permission
         SPACE
         CALL  BPX1UMK,              Set file creation mask            +
               (S_MODE,              Input: Mode              BPXYMODE +
               RETVAL),              Return value: previous mode mask  +
               VL,MF=(E,PLIST)       ----------------------------------
```

Related Services

Other callable services related to this service are:

- "mkdir (BPX1MKD) — Make a Directory" on page 160
- "open (BPX1OPN) — Open a File" on page 181.

umount (BPX1UMT) — Remove a Virtual File System

BPX1UMT

file_system_name_length
file_system_name
flags
return_value
return_code
reason_code

Purpose

Use the umount (BPX1UMT) service to unmount a virtual file system (remove the virtual file system from the file tree).

Parameters

file_system_name_length
(input,INT,4) is a variable for specifying the length of the *file_system_name* parameter.

file_system_name
(input,CHAR,file_system_name_length) is a variable for a printable-character field that contains the name of the file system to be unmounted. The name must be left-justified and padded with blanks. The file system name can be a Byte File System (BFS) path name or a Network File System (NFS) path name. See usage note "2" on page 375.

flags
(input,INT,4) is a variable for a field containing binary flags that specify the unmount options. This field is mapped by the BPXYMTM macro. See "BPXYMTM — Map the Modes for the mount and umount Services" on page 445.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. When a file system is unmounted, all file systems mounted below it in the hierarchy are also unmounted.
2. The *file_system_name* can represent a BFS path name or an NFS path name:
   - To unmount a BFS file system, *file_system_name* must be a BFS path name. See "Understanding Byte File System (BFS) Path Name Syntax" on page 6.
   - To unmount an NFS file system, *file_system_name* must be a fully-qualified NFS path name. See "Understanding Network File System (NFS) Path Name Syntax" on page 9.

Example

The following code removes the virtual file system previously mounted at directory /u from the file tree. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structure, see "BPXYMTM — Map the Modes for the mount and umount Services" on page 445.

```
         LA    R6,2
         ST    R6,LFSNAME
         MVC   FSNAME(2),=CL02'/u'
         XC    MTM(MTM#LENGTH),MTM
         MVI   MTM1,MTMUMOUNT        Unmount request
         SPACE ,
         CALL  BPX1UMT,              Remove a virtual file system      +
               (LFSNAME,             Input: File system name length    +
               FSNAME,               Input: File system name           +
               MTM,                  Input: Flags, BPXYMTM             +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return code:

Return Code	Explanation
EINVAL	An incorrect parameter was specified. The *file_system_name* value is not the name of a mounted file system. Consult the reason code to determine the exact reason the error occurred.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, "Return Codes," on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, "Reason Codes," on page 495.

Related Service

Another callable service related to this service is:

- "mount (BPX1MNT) — Make a File System Available" on page 166

uname (BPX1UNA) — Display the Name of the Current Operating System

BPX1UNA

data_area_length
data_area_address
return_value
return_code
reason_code

Purpose

Use the uname (BPX1UNA) service to obtain information about the OpenExtensions system you are running on.

Parameters

data_area_length
(input,INT,4) is a variable for specifying the length of the data area pointed to by the *data_area_address* parameter. The area must be at least the size specified in the UTSN#LENGTH field of the BPXYUTSN macro. See "BPXYUTSN — Map the System Information Structure for the uname Service" on page 480.

data_area_address
(input,INT,4) is a variable for specifying the address of the buffer where the service is to return the system information. This data area is mapped by the BPXYUTSN macro. See "BPXYUTSN — Map the System Information Structure for the uname Service" on page 480.

return_value
(output,INT,4) is a variable where the service returns a nonnegative value if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Example

The following code obtains information about the system on which the invoker is running. This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structure, see "BPXYUTSN — Map the System Information Structure for the uname Service" on page 480.

```
         LA    R15,UTSN
         ST    R15,UTSNA
         LA    R15,UTSN#LENGTH
         ST    R15,UTSNL
         SPACE
         CALL  BPX1UNA,             Identify system                  +
               (UTSNL,              Input: Length of required buffer  +
               UTSNA,               Output: ->UTSN           BPXYUTSN +
               RETVAL,              Return value: -1 or >-1           +
               RETCODE,             Return code                       +
```

```
                RSNCODE),             Reason code                        +
                VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECMSERR	A CMS environmental or internal error occurred. Consult the reason code to determine the exact reason the error occurred. The following reason codes can accompany this return code: JrIdentifyErr, JrStackReadErr, and JrQEFLErr.
EINVAL	The passed length of the invoker UTSN is not valid. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JROK.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

unlink (BPX1UNL) — Remove a Directory Entry

BPX1UNL

name_length
name
return_value
return_code
reason_code

Purpose

Use the unlink (BPX1UNL) service to remove a directory entry. A directory entry could be identified by a path name to a file, a link name to a file, or a symbolic link.

If a link to a file is removed, and the link count becomes zero, and no other process has the file open, the file itself is deleted.

Parameters

name_length
(input,INT,4) is a variable for specifying the length of the *name* parameter.

name
(input,CHAR,*name_length*) is a variable for specifying the name of the directory entry to be removed. This name can be a path name to a file, a link name to a file, or a symbolic link name. The path name was specified when the file was created. (See "open (BPX1OPN) — Open a File" on page 181.) The link name was specified when a link to the file was created or when the symbolic link was created. (See "link (BPX1LNK) — Create a Link to a File" on page 149 or "symlink (BPX1SYM) — Create a Symbolic Link to a Path Name" on page 345.)

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. If *name* refers to a symbolic link, then the symbolic link file named by *name* is deleted.
2. If a file is deleted—that is, if the unlink service request is successful and the link count becomes zero—the file is deleted. The contents of the file are discarded, and the space it occupied is freed for reuse. However, if another process (or more than one) has the file open when the last link is removed, the file is not removed until the last process closes it.
3. When the unlink (BPX1UNL) service is successful in removing the directory entry and decrementing the link count, whether or not the link count becomes zero, it returns control to the caller with *return_value* set to `0`. It updates the change and modification times for the parent directory, and the change time for the file itself (unless the file is deleted).
4. Directories cannot be removed using unlink (BPX1UNL). To remove a directory, refer to "rmdir (BPX1RMD) — Remove a Directory" on page 256.

Example

The following code removes path name **usr/dataproc/next.t** from the system. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   BUFFERA(19),=CL19'usr/dataproc/next.t'
         MVC   BUFLENA,=F'19'
         SPACE ,
         CALL  BPX1UNL,              Remove a directory entry          +
               (BUFLENA,             Input: Pathname length            +
               BUFFERA,              Input: Pathname                   +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	The calling process does not have permission to search some component of the path name, or did not have write permission for the directory containing the link to be removed.
EBUSY	The file cannot be unlinked because it is being used by the system.
EINVAL	The *name* parameter is incorrect. It contains a null character.
ELOOP	A loop exists in symbolic links encountered during resolution of the *name* argument. This error is issued if more than 8 symbolic links are detected in the resolution of *name*.
ENAMETOOLONG	The *name* argument is longer than 1023 characters, or some component of the name is longer than 255 characters. CMS does not support name truncation.
ENOENT	The *name* entry was not found, or no name was specified. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRUnlNoEnt.
ENOTDIR	Some component of the path name prefix is not a directory.
EPERM	The *name* argument refers to a directory. Directories cannot be removed using this service. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRUnlDir.
EROFS	The link to be removed is on a read-only file system. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRUnlMountRO.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “close (BPX1CLO) — Close a File or Socket” on page 34

- “link (BPX1LNK) — Create a Link to a File” on page 149
- “open (BPX1OPN) — Open a File” on page 181
- “rename (BPX1REN) — Rename a File or Directory” on page 251
- “rmdir (BPX1RMD) — Remove a Directory” on page 256.

utime (BPX1UTI) -- Set File Access and Modification Times

```
BPX1UTI
        pathname_length
        pathname
        newtimes
        return_value
        return_code
        reason_code
```

Purpose

Use the utime (BPX1UTI) service to set the access and modification times of a file.

Parameters

pathname_length
(input,INT,4) is a variable for specifying the length of the *pathname* parameter.

pathname
(input,CHAR,*pathname_length*) is a variable for specifying the path name of the file. See "Understanding Byte File System (BFS) Path Name Syntax" on page 6.

newtimes
(input,CHAR,8) is a variable for specifying the access and modification times for the file. The first fullword contains the new access time, and the second fullword contains the new modification time. These times can be retrieved with "stat (BPX1STA) -- Get Status Information about a File by Path Name" on page 340 or "fstat (BPX1FST) -- Get Status Information about a File by Descriptor" on page 102.

- Times are specified in POSIX format, which is the number of seconds since January 1, AD 1970, 00:00:00 UTC. The times must be specified as nonnegative values other than -1 (see below for the special case of -1).
- To request that the current time be used for both access and modification times, specify X'FFFFFFFF' (-1) in either or both words of this field. The current time in the file's status is also updated.

return_value
(input,INT,4) is a variable where the service returns 0 if the request is successful, or -1 if it is not successful.

return_code
(input,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Note

If you need to perform conversions on POSIX times, see the DateTimeSubtract CSL routine in the *z/VM: CMS Application Multitasking* or the DATECONVERT stage in the *z/VM: CMS Pipelines User's Guide and Reference*.

Example

The following code changes the access and modification times of **/usr/private/workfile.t** to the current time. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   BUFFERA(23),=CL23'/usr/private/workfile.t'
         MVC   BUFLENA,=F'23'
         MVC   NEWTIMES,=D'-1'       Current time
         SPACE ,
         CALL  BPX1UTI,              Set file access and modify times  +
               (BUFLENA,             Input: Pathname length            +
               BUFFERA,              Input: Pathname                   +
               NEWTIMES,             Input: Access & Modification time +
               RETVAL,               Return value: 0 or -1             +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	One of the following is true: • The process does not have search permission for some component of the path name prefix. • The *newtimes* value equals the current time, the effective ID does not match the file's owner, the process does not have write permission for the file, and the process does not have appropriate privileges.
EINVAL	The argument supplied is incorrect. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRNegativeValueInvalid.
ELOOP	A loop exists in symbolic links encountered during resolution of the *pathname* argument. This error is issued if more than 8 symbolic links are detected in the resolution of the path name.
ENAMETOOLONG	The length of *pathname* is greater than 1023 bytes, or some component of the fully qualified name is longer than 255 bytes. This could be as a result of encountering a symbolic link during resolution of the path name, and the substituted string is longer than 1023 characters.
ENODEV	An attempt was made to use a character special file for a device not supported by OpenExtensions.
ENOENT	No file named *pathname* was found, or the *pathname* parameter was blank. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRFileNotThere.
ENOTDIR	Some component of the path name prefix is not a directory.
EPERM	The *newtimes* value did not specify the current time, the effective user ID of the calling process does not match the owner of the file, and the calling process does not have appropriate privileges.
EROFS	The *pathname* file is on a read-only file system. Consult the reason code to determine the exact reason the error occurred. The following reason code can accompany this return code: JRReadOnlyFs.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “fstat (BPX1FST) -- Get Status Information about a File by Descriptor” on page 102
- “stat (BPX1STA) -- Get Status Information about a File by Path Name” on page 340.

wait (BPX1WAT) — Wait for a Child Process to End

BPX1WAT

process_ID
options
status_field_address
return_value
return_code
reason_code

Purpose

Use the wait (BPX1WAT) service to obtain the status of a child process that has ended or stopped. The term *child* refers to a process created by the spawn (BPX1SPN) service.

Parameters

process_ID
(input,INT,4) is a variable for specifying a value that indicates the event to be waited on:

- A value greater than zero is assumed to be a process ID. The caller waits for the child with that specific process ID to end or stop.
- A value of zero indicates the caller is waiting for any children with a process group ID equal to the caller's to end or stop.
- A value of -1 indicates the caller is waiting for any of its children to end or stop.
- If the value is negative and less than -1, its absolute value is assumed to be a process group ID. The caller waits for any children with that process group ID to end or stop.

options
(input,INT,4) is a variable for specifying the wait options for this invocation. These options affect the actions taken by the service as described below. The options can be specified separately or in combination. A zero value for this parameter implies that the service performs its default processing—that is, it waits for a child process to end or stop.

The following flags defined in the BPXYCONS macro are the allowed wait options. See "BPXYCONS — Map Constants" on page 417.

Constant	Description
WNOHANG	The service does not suspend execution of the calling process if status is not immediately available for one of the child processes specified by *process_ID*.
WUNTRACED	The service also returns the status of any child processes specified by *process_ID* that are stopped, and whose status have not yet been reported since they stopped.

status_field_address
(input,INT,4) is a variable for specifying the address of a fullword where the service returns the status value for the child process that ended or stopped. The status value can be analyzed using the BPXYWAST macro. See "BPXYWAST — Map the Wait Status Word" on page 486. The status value is returned only if status is available for a child process.

return_value
(output,INT,4) is a variable where the service returns the process ID of the child the status information applied to if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

The wait (BPX1WAT) service suspends execution of the calling thread until one of the requested child processes ends or until it obtains information about the process. If a child has already ended but its status has not been reported when wait (BPX1WAT) is called, the routine immediately returns with that status information to the caller.

The wait service always returns status for the stopped processes, even if WUNTRACED is not specified.

If status is available for one or more processes, the order the status is reported is unspecified.

If the wait (BPX1WAT) service is invoked simultaneously from multiple threads within the same process, the following behavior should be noted:

- When multiple threads issue a spawn call followed by a call to the wait (BPX1WAT) service to wait for any child process to end, the status received by each thread may not be the status of the child created by that thread. If a thread wishes to receive the status of the child it created, the thread should specify the returned child Process Id when calling the wait (BPX1WAT) service to wait for the child process to end or stop.
- If the wait (BPX1WAT) service is called from multiple threads requesting status for the same process, which thread receives the status is not specified. The thread that does not receive the status is returned to with a return value of -1 and a return code of ECHILD.

Example

The following code waits for any of its children to end or stop. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551. For the data structure, see “BPXYWAST — Map the Wait Status Word” on page 486 and “BPXYCONS — Map Constants” on page 417.

```
         LA    R15,WAST             Resolve address of STATUS
         ST    R15,WASTA            Save address of STATUS
         MVC   PROCID,=F'-1'        Wait for any child
         SPACE ,
         CALL  BPX1WAT,             Wait for a child process to end   +
               (PROCID,             Input: PID being waited on        +
               =A(WNOHANG),         Input: options       BPXYCONS     +
               WASTA,               ->Exit status field, BPXTWAST     +
               RETVAL,              Return value: -1, 0, child PID    +
               RETCODE,             Return code                       +
               RSNCODE),            Reason code                       +
               VL,MF=(E,PLIST)      ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECHILD	The caller has no appropriate child process; that is, no child process whose status has not already been obtained through earlier calls to the wait (BPX1WAT) service meets the criteria for waiting.

Return Code	Explanation
EINTR	The calling process received a signal prior to the completion of an event that would cause the wait (BPX1WAT) service to return. The service was interrupted by a signal. In this case, the value contained in *status_field_address* is undefined.
EINVAL	The value of the option is not valid.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “_exit (BPX1EXI) — End a Process and Bypass the Cleanup” on page 79
- “pause (BPX1PAS) — Suspend a Process Pending a Signal” on page 197
- “spawn (BPX1SPN) — Spawn a Process” on page 333.

wait-extension (BPX1WTE) — Obtain Status Information for Child Processes

BPX1WTE
function_code
ID_type
ID
stat_loc_ptr
options
info_area_ptr
return_value
return_code
reason_code

Purpose

Use the wait-extension (BPX1WTE) service to obtain status information about child processes of the parent that calls the routine.

Parameters

function_code
(input,INT,4) is a variable for specifying a value that indicates the function to be performed:

Value
Description

#WAITID
The waitid() function is performed.

The #WAITID constant is defined in the BPXYCONS macro. See "BPXYCONS — Map Constants" on page 417.

ID_type
(input,INT,4) is a variable for specifying a value that indicates what type of child processes to wait for. The *ID_type* can be one of the following values:

Value
Description

P_PID
The waitid() function will wait for the child process whose process ID is equal to the value specified in the *id* parameter.

P_PGID
The waitid() function will wait for the child processes whose process group ID is equal to the value specified in the *id* parameter.

P_ALL
The waitid() function will wait for all child processes. The *ID* parameter is ignored.

The P_ constants are defined in the BPXYCONS macro.

ID
(input,INT,4) is a variable for specifying the process ID or process group ID of the child processes to wait for. Together with *ID_type*, *ID* is used to determine which child processes will be waited for.

stat_loc_ptr
(input,INT,4) is a variable for specifying the address of a fullword where this service can place the status value (wait status word) for the child process, if status is available. This parameter is not valid for the #WAITID function code and is ignored.

options
(input,INT,4) is a variable for specifying the wait options for this call:

- For function code #WAITID, this parameter specifies which state changes to wait for:

  Option
  Description

  WEXITED
  Wait for child processes that have exited.

  WSTOPPED
  Status will be returned for any child that has stopped upon receipt of a signal.

  WCONTINUED
  Status will be returned for any child that has stopped and has been continued.

  WHOHANG
  Return immediately if there are no children to wait for.

  WHOWAIT
  Keep the process whose status is returned in the *info_area_ptr* parameter in a waitable state. This will not affect the state of the process; the process may be waited for again after this call completes.

These option constants are defined in the BPXYCONS macro.

info_area_ptr
(input,INT,4) is a variable for specifying the address where the service returns information into a data structure:

- For function code #WAITID, this is the address of a Siginfo_t structure. The Siginfo_t structure type is defined in the BPXYSINF macro. See "BPXYSINF — Map the Siginfo_t Structure for the wait-extensions Service" on page 464. If this field is null, no information is returned.

return_value
(output,INT,4) is a variable where the service returns the following value if the request is successful:

- For function code #WAITID, the service returns 0.

If the request is not successful, the service returns -1.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. When the Siginfo_t structure is returned, the following applies:
   - SI_SIGNO is always set to SIGCHLD.
   - SI_ERRNO is always set to 0.
   - SI_CODE is set to CLD_EXITED, CLD_KILLED, CLD_DUMPED, CLD_TRAPPED, CLD_STOPPED, or CLD_CONTINUED. The CLD_ constants are defined in the BPXYSIGH macro. See "BPXYSIGH — Map Signal Constants" on page 462.
   - SI_PID is set to the process ID of the child status is being returned for.
   - SI_UID is set to the user ID of the child status is being returned for.

- SI_ADDR is set to the faulting instruction if the child process terminated because of a SIGILL, SIGFPE, or SIGSEGV signal; otherwise, SI_ADDR is set to 0.
- SI_STATUS is set to the child's exit status. The exit status is mapped by the BPXYWAST macro.
- SI_BAND is always set to 0.

2. If the *options* parameter is set to 0, the wait-extension (BPX1WTE) service waits for processes that have exited.

Example

The following code uses the #WAITID function to wait for any of its children to end or stop. For the data structures, see "BPXYWAST — Map the Wait Status Word" on page 486, "BPXYSINF — Map the Siginfo_t Structure for the wait-extensions Service" on page 464, "BPXYCONS — Map Constants" on page 417, and "BPXYSIGH — Map Signal Constants" on page 462.

```
         LA    R15,WAST              Resolve address of WAST
         ST    R15,WASTA             Save address of WAST
         LA    R15,SIGINFO_T         Resolve address of SIGINFO_T
         ST    R15,SIGINFO_TA        Save address of SIGINFO_T
         SPACE ,
         CALL  BPX1WTE,              Wait for a child process to end   +
               (=A(#WAITID),         Input: function      BPXYCONS     +
               P_ALL,                Input: id type (any child)        +
               0,                    Input: id                         +
               WASTA,                ->Exit status field, BPXYWAST     +
               =A(WNOHANG),          Input: options       BPXYCONS     +
               SIGINFO_T,            ->Siginfo structure, BPXYSINF     +
               RETVAL,               Return value: -1, 0, child PID    +
               RETCODE,              Return code                       +
               RSNCODE);             Reason code                       +
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
ECHILD	The calling process has no existing unwaited-for child processes.
EFAULT	The address of a returned parameter is not valid. The following reason codes can accompany this return code: JRBadExitStatusAddr, JRBadSiginfoAddr, or JRBadRusageAddr.
EINTR	The function was interrupted because of the receipt of a signal by the calling process.
EINVAL	The specified *option*, *idtype*, or *function_code* was not valid. The following reason codes can accompany this return code: JRBadOptions, JRBadIdType, or JRBadEntryCode.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, "Return Codes," on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, "Reason Codes," on page 495.

w_getipc (BPX1GET) — Query Interprocess Communications

```
BPX1GET
        token_or_ID
        buffer_address
        buffer_length
        command
        return_value
        return_code
        reason_code
```

Purpose

Use the w_getipc (BPX1GET) service to query message queues, shared memory segments, and semaphore sets for a specified member or the next member to which the caller has read access.

Parameters

token_or_ID
(input,INT,4) is a variable for specifying one of the following:

- A token that corresponds to one member of a group that includes all the message queues, shared memory segments, and semaphore sets to which the caller has read access. That is, the member is a message queue, shared memory segment, or semaphore set in the group. (The *command* parameter indicates whether all of the members of the group are to be queried, or only the message queues, or only the shared memory segments, or only the semaphore sets.) A token of `0` represents the first member of the group to be queried. The token to be used in the next invocation of this service (to query the next member of the group) is passed back in the *return_value* parameter.
- The identifier of a specific message queue, shared memory segment, or semaphore set to be queried.

This parameter is ignored when the IPC_OVER command is specified.

buffer_address
(input,INT,4) is a variable for specifying the address of a buffer mapped by the IPCQ data structure in the BPXYIPCQ macro. See "BPXYIPCQ — Map the Data Structure and Constants for the w_getipc Service" on page 432.

buffer_length
(input,INT,4) is a variable for specifying the size of the buffer pointed to by the *buffer_address* parameter. This is set to IPCQ#LENGTH, which is defined in IPCQ. The IPCQLENGTH field of IPCQ will differ from IPCQ#LENGTH when the system call is at a different level than the included IPCQ. An error is returned if *buffer_length* is less than 4. The buffer will be filled to the lesser of IPCQ#LENGTH or the value specified here.

command
(input,INT,4) is a variable for specifying a command constant that identifies the type of query to be performed. The following command constants are defined in the BPXYIPCQ macro:

IPCQ#ALL
Retrieves data about the next message queue, shared memory segment, or semaphore set from the group.

IPCQ#MSG
Retrieves data about the next message queue from the group.

IPCQ#SEM
Retrieves data about the next semaphore set from the group.

IPCQ#SHM
Retrieves data about the next shared memory segment from the group.

IPCQ#OVER
Obtains an overview of system variables. When this command is specified, the *token_or_ID* parameter is ignored.

return_value
(output,INT,4) is a variable where:

- If a token is specified in the *token_or_ID* parameter, the service returns one of the following:
  - The token (a negative number other than -1) for the next member of the group to be queried
  - A value of 0, which indicates end of file (no more members to be queried)
  - A value of -1, which indicates the request failed
- If an identifier is specified in the *token_or_ID* parameter, the service returns a value of 0 if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. If a token is specified for *token_or_ID*, the *return_value* should be tested for 0 (end of file) or -1 (error). Any other value is negative and is the token to be used in the next invocation of the service.
2. If an identifier is specified for *token_or_ID*, the *return_value* should be tested for -1 (error).
3. A member's accessibility can change if the permissions are changed.
4. A token may not always retrieve the same member. If a specific member has been found by using a token, subsequent requests may place the member at that token or later, but never earlier.

Characteristics and Restrictions

There are no restrictions on the use of the w_getipc service.

Example

The following code retrieves information on the first semaphore defined to the system to which the caller has read access. For the data structure, see "BPXYIPCQ — Map the Data Structure and Constants for the w_getipc Service" on page 432.

```
         XC    TOKEN,TOKEN           Zero, token for 1st member
         LA    R5,BUFFERA            Area for query IPC return data
         ST    R5,BUFA               R5 -> IPCQ
         SPACE ,
         CALL  BPX1GET,              Interprocess Communications       +
               (TOKEN,               Input: member token                +
               BUFA,                 Input: ->IPCQ             BPXYIPCQ+
               =A(IPCQ#LENGTH),      Input: Length of IPCQ     BPXYIPCQ+
               =A(IPCQ#SEM),         Input: Request            BPXYIPCQ+
               RETVAL,               Return value: 0, -1 or value      +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
         SPACE ,
         L     R15,RETVAL            Load return value
         C     R15,=F'-1'            Test for -1 return
         BE    PSEUDO                Branch on error
```

```
         LTR   R15,R15              Test for 0 return
         BZ    PSEUDO               Branch on end of file
         ST    R15,TOKEN            Save token for next w_semipc
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EACCES	Operation permission (read) is denied to the calling process for the specified message queue identifier, shared memory segment identifier, or semaphore set identifier. The following reason code can accompany this return code: JRIpcDenied.
EINVAL	One of the following conditions is true: • The specified message queue identifier, shared memory segment identifier, or semaphore set identifier is not valid for the specified command. • *command* is not a valid command. • *buffer_address* is zero, or *buffer_length* is less than 4. The following reason codes can accompany this return code: JRBuffTooSmall, JRIpcBadID, JRBadEntryCode.
EFAULT	An input parameter specified an address that caused the callable service to program check. The following reason code can accompany this return code: JRBadAddress.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “msgget (BPX1QGT) — Create or Find a Message Queue” on page 172
- “semget (BPX1SGT) — Create or Find a Set of Semaphores” on page 269
- “shmget (BPX1MGT) — Create or Find a Shared Memory Segment” on page 309

w_getpsent (BPX1GPS) -- Get Process Data

BPX1GPS

process_token
buffer_length
buffer_address
return_value
return_code
reason_code

Purpose

Use the w_getpsent (BPX1GPS) service to get data describing the status of a process. This data includes, but is not limited to, running time, user IDs (UIDs), groups IDs (GIDs), and invocation parameters. Data is returned for the processes the caller can access.

Parameters

process_token
(input,INT,4) is a variable for specifying a process token that identifies the relative position of a process in the system. Zero represents the first process in the system.

buffer_length
(input,INT,4) is a variable for specifying the size of the buffer, which is specified in the PGPS#LENGTH field of the BPXYPGPS macro.

buffer_address
(input,INT,4) is a variable for specifying the address of the buffer where the service is to return the process data. These options are mapped by the BPXYPGPS macro. See "BPXYPGPS — Map the Response Structure for the w_getpsent Service" on page 449. Several fields in this buffer should be initialized as follows:

```
PGPSCONTTYBLEN    Length of PGPSCONTTYBUF
PGPSCONTTYPTR     Address of PGPSCONTTYBUF(Len¬=0)
PGPSPATHBLEN      Length of PGPSPATHBUF
PGPSPATHPTR       Address of PGPSPATHBUF (Len¬=0)
PGPSCMDBLEN       Length of PGPSCMDBUF
PGPSCMDPTR        Address of PGPSCMDBUF (Len¬=0)
```

return_value
(output,INT,4) is a variable where the service returns one of the following values:

Value	Explanation
Process Token	The process token of the next logical process in the system.
`0`	End of file. There are no active processes at or following the requested process which the user is allowed access.
`-1`	Error. See the return code for an explanation.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. Information is returned only for processes in the caller's virtual machine.
2. Generally, the user starts with *process_token* at zero, and continues calling the w_getpsent (BPX1GPS) service with the process token returned as the *return_value* of the previous call until the value of 0, end of file, is reached.
3. The PGPSSTARTTIME field in the buffer is in POSIX format, which is the number of seconds since January 1, AD 1970, 00:00:00 UTC. If you need to perform conversions on POSIX times, see the DateTimeSubtract CSL routine in the *z/VM: CMS Application Multitasking* or the DATECONVERT stage in the *z/VM: CMS Pipelines User's Guide and Reference*.
4. PGPSUSERTIME and PGPSSYSTIME are task-elapsed times in 1/100ths of seconds.
5. The CONTTY, PATH, and CMD input fields are initialized by the BPXYPGPS macro when it is expanded in the program CSECT for a non-reentrant program.
6. If *buffer_length* does not match that used by the callable service, the service sets PGPSLENERR on. This can reflect a change in BPXYPGPS caused by the addition of functions in later releases. This could be intentional. Data is returned up to the length specified in *buffer_length*. If the length specified is less than the offset of PGPSCONTTYBLEN, BPX1GPS treats the request as if the three BLEN fields were zero.

Example

The following example starts with the first process (relative process zero) and reports the status for all processes for which the invoker is allowed access (by the security access facility).

This example follows the rules of reentrancy. For a nonreentrant example of this service, see "Nonreentrant Entry Linkage" on page 553.

```
BOOKSAM4 CSECT ,                    Reentrant linkage
BOOKSAM4 AMODE 31
BOOKSAM4 RMODE ANY
         USING *,R15                Program addressability
@BEGIN0  B     @BEGIN1              Branch around program header
         DROP  R15
         DC    C'Sequential w_getpsent'
         DS    0H
@BEGIN1  STM   R14,12,12(13)        Save caller's registers
         LR    R2,13                Hold address of caller's area
         LR    R3,R1                Hold parameter register
         LR    12,R15               R12 program base register
         USING @BEGIN0,12           Program addressability
         L     R0,@SIZEDAT          Size this program's dynamic area
         GETMAIN RU,LV=(0)          Getmain dynamic storage
         LR    13,R1                R13 -> this program's dynamic/save
         USING @DYNAM,13            Dynamic addressability
         ST    R2,@BACK             Save caller's save area pointer
         ST    13,8(,R2)            Give caller out save area
         LR    R1,R3                Restore parameter register
@BEGIN2  EQU   *  *  *  *  *  *  *   End of the entry linkage code
         SPACE ,
         MVC   WTOHEAD,WTOCONS      Initialize WTO line
         MVI   DOT,C'.'
* If BPX1GPS has been link-edited with this program, the V-CON will be
* resolved; if not, BPX1GPS must be loaded.  In either case, the address
* of the module is stored.
         ICM   R0,B'1111',GPSVCON   BPX1GPS address if link edited
         BNZ   STGPSEP              Branch to store GPS entry point
         LOAD  EP=BPX1GPS           Load w_getpsent stub
STGPSEP  ST    R0,GPSENTRY          Store BPX1GPS entry point
* Initialize the variables and enter the loop.
         XC    PROCTOKEN,PROCTOKEN  Start with 1st process
         MVC   PGPSCONTTYBLEN,=A(L'PGPSCONTTYBUF)  Controlling TTY
         LA    R2,PGPSCONTTYBUF
         ST    R2,PGPSCONTTYPTR
         MVC   PGPSPATHBLEN,=A(L'PGPSPATHBUF)      Path name
         LA    R2,PGPSPATHBUF
         ST    R2,PGPSPATHPTR
         MVC   PGPSCMDBLEN,=A(L'PGPSCMDBUF)        Command
         LA    R2,PGPSCMDBUF
         ST    R2,PGPSCMDPTR
         LA    R2,PGPS              Address of PGPS buffer
```

```
         ST    R2,PGPSA
         SPACE ,
GETPS    L     R15,GPSENTRY          Address of BPX1GPS load module
         CALL  (15),                 Get process data                  +
               (PROCTOKEN,           Relative process token            +
               PGPSL,                Length of buffer                  +
               PGPSA,                Buffer, mapped by BPXYPGPS        +
               RETVAL,               Return value (next, eof or error) +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ---------------------------------
         SPACE ,  *  *  *  *  *  *   Test for end of file
         ICM   R2,B'1111',RETVAL     Load return value, set CCode
         BZ    RETURN                0 is end of file
         BL    RETURNRC              -1 is error
         ST    R2,PROCTOKEN          Store the next process token
         SPACE ,  *  *  *  *  *  *   Initialize WTO area and message
         MVI   XPID,C' '             Blank variable portion of line
         MVC   XPID+1(WTO#BLANK-1),XPID
* Convert the process ID to printable hex.
         L     R8,PGPSPID            R8 = process ID
         LA    R9,XPID               To be placed at message start
         LA    R15,8                 8 nibbles to convert (4 bytes)
         LA    R10,9                 For 0-9 / A-F compare
NIBBLE   LR    R11,R8                Target bits in 0-3     XYYYYYYZ
         SRL   R11,28                Bits 0-3 to 28-31      0000000X
         SLL   R8,4                  Drop bits 0-3 off end  YYYYYYZ0
         CLR   R11,R10               Are 4 bits 0-9 or A-F
         BC    B'0010',AF            Branch if A-F
         LA    R11,57(,R11)          Add for 0-9 (57+183=240 or F0)
AF       LA    R11,183(,R11)         Add for 0-F (183+10=193 or C1)
         STC   R11,0(,R9)            Store to results location
         LA    R9,1(,R9)             Increment R9 to next location
         BCT   R15,NIBBLE            Decrement half byte counter, loop
* Go after the state of the process
         MVI   THREAD,C'1'                Assume single task thread
         TM    PGPSSTATUS1,PGPSMULTHREAD  if multithread process
         BZ    NOTMULT
         MVI   THREAD,C'M'
NOTMULT  TM    PGPSSTATUS1,PGPSPTHREAD    if pthread_create task(s)
         BZ    NOTIPT
         MVI   THREAD,C'H'
NOTIPT   MVC   STATE,PGPSSTATUS3          Z, W, X, S, C, F, K, R
         TM    PGPSSTATUS0,PGPSSWAP       if swapped out
         BZ    NOTSWAP
         MVC   SWAPA,=CL4'SWAP'
NOTSWAP  TM    PGPSSTATUS1,PGPSSTOPPED    if stopped
         BZ    NOTSTOP
         MVC   STOPA,=CL4'STOP'
NOTSTOP  EQU   *
         SPACE ,  *  *  *  *  *  *   Display message to operator
         WTO   MF=(E,WTOAREA)        Write to Operator
         SPACE ,  *  *  *  *  *  *   Loop back
         B     GETPS                 for the next Process data
         SPACE ,
* * * * *.* * *.* * * * * * * * * * *.* * * * * * * * * * * * * * * * * *
RETURN   XR    R15,R15               Zero return code
RETURNRC L     R0,@SIZEDAT           Size this program's dynamic area
         LR    R1,13                 R1 -> this program's dynamic area
         L     13,@BACK              R2 -> caller's save area
         DROP  13
         FREEMAIN RU,LV=(0),A=(1)
         L     R14,12(,13)           Restore caller's R14
         LM    R0,12,20(13)          Restore caller's R0-R12
         BSM   0,R14                 Branch back to caller
@SIZEDAT DC    A(@ENDYN-@DYNAM)      Size of dynamic storage
         SPACE ,  * * * * * * * * *.* Program constants * * * * * * *
PGPSL    DC    A(PGPS#LENGTH)        Length of process data buffer
         WXTRN BPX1GPS               Weak to allow link edit or not
GPSVCON  DC    V(BPX1GPS)            Get Process data module
WTOCONS  DS    0CL8                  Constant value for WTOHEAD
         DC    AL2(WTO#LENGTH)         Length of area
         DC    AL2(0)                  WTO flags
         DC    CL4'PID='               Process ID =
         SPACE ,  *  *  *  *  *  *   Dynamic storage variables
@DYNAM   DSECT ,
@SAVE00  DS    0D                    Standard save area - 72 Bytes
         DS    A
@BACK    DS    A                     Backwards savearea pointer
@FORWARD DS    A                     Forwards savearea pointer
         DS    15A                   Regs 14,15,0-12
         SPACE ,
```

```
WTOAREA  DS    0F                  WTO message
WTOHEAD  DS    CL8                 Mapped by WTOCONS
XPID     DS    CL8                 Hex of process ID
         DS    CL1
THREAD   DS    CL1                 1, M or H
         DS    CL1
STATE    DS    CL1                 Z, W, X, S, C, F, K, R
         DS    CL1
SWAPA    DS    CL4                 SWAP or blank
         DS    CL1
STOPA    DS    CL4                 STOP or blank
         DS    CL1
TRACA    DS    CL4                 TRAC or blank
WTO#BLANK EQU   *-XPID             Length to blank
DOT      DS    CL1
WTO#LENGTH EQU *-WTOAREA           Length of WTO area
         SPACE ,
GPSENTRY DS    A                   Address of BPX1GPS
PROCTOKEN DS   F                   Relative process token
PLIST    DS    6A                  Calling parameter list
RETVAL   DS    F                   Return value - next PROCTOKEN
RETCODE  DS    F                   Return code
RSNCODE  DS    F                   Reason code
         SPACE ,
PGPSA    DC    A(PGPS)             ->Process data buffer
         BPXYPGPS DSECT=NO,        Place in current dsect           +
               VARLEN=(0,0,0)      ConTty=0,Path=0,Cmd=0
@ENDYN   EQU   *                   End of dynamic storage
         SPACE 3  * * * * * * * * * * *.* Register equates  * * * * * * *
R0       EQU   0
R1       EQU   1                   Parameter list pointer
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
*        EQU   12                  Program base register
*        EQU   13                  Savearea and dynamic storage base
R14      EQU   14                  Return address
R15      EQU   15                  Branch location
         SPACE ,
         END
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EFAULT	An input parameter contained the address of storage where the invoker is not authorized.
EINVAL	The specified *process_token* is not in the valid range.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

w_ioctl (BPX1IOC) — Control I/O

BPX1IOC

file_descriptor
command
argument_length
argument
return_value
return_code
reason_code

Purpose

Use the w_ioctl (BPX1IOC) service to convey a command to a device. The specific actions performed by this service vary by device, and are defined by the device driver.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the descriptor of an open file or socket.

command
(input,INT,4) is a variable for specifying the ioctl command to be passed to the device driver. The values for this field are defined in the BPXYIOCC macro. See "BPXYIOCC — Map Command Constants for the w_ioctl Service" on page 427.

argument_length
(input/output,INT,4) is a variable for specifying the length of the *argument* parameter. This value must be an integer in the range 0–1024. On return from w_ioctl, the service updates this field with the length of the command output returned in the *argument* parameter.

argument
(input/output,INT,*argument_length*) is a variable for specifying the argument to be passed to the device driver. On return from w_ioctl, the service updates this field with the command output, if any.

return_value
(output,INT,4) is a variable where the service returns `0` if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

1. AF_UNIX domain sockets support the following commands:
   - FIONBIO
   - FIONREAD
   - SECIGET
   - SIOCATMARK

2. AF_INET and AF_INET6 sockets pass the ioctl command to TCP/IP. For the commands supported, refer to the *XL C/C++ for z/VM: Runtime Library Reference*.
3. Remote terminals support the TIOCGWINSZ and TIOCSWINSZ command to get and set the window size.
4. The pipe file system does not support ioctl.

Characteristics and Restrictions

The argument is limited to 1024 bytes.

Example

The following code conveys a command to the standard output device. To run properly, this example needs a command defined by the user for the COMMAND parameter. This command must be understood by the device driver providing support for the output device. This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
         MVC   BUFLENA,=F'1024'
         MVC   COMMAND,=F'123'       User defined command
         SPACE ,
         CALL  BPX1IOC,              I/O Control                      +
               (=A(STDOUT_FILENO),   Input: File descriptor           +
               COMMAND,              Input: Command                   +
               BUFLENA,              Input: Argument length           +
               BUFFERA,              Argument buffer name             +
               RETVAL,               Return value: 0 or -1            +
               RETCODE,              Return code                      +
               RSNCODE),             Reason code                      +
               VL,MF=(E,PLIST)       ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EAFNOSUPPORT	The address family is not supported.
EALREADY	An attempt was made to unregister a file that is not registered.
EBADF	*file_descriptor* is not a valid socket descriptor.
EINVAL	One of the following occurred: • *argument_length* was not valid. The correct argument length range is 0 to 1024. • *command* was not valid. The following reason codes can accompany this return code: JRInvIoctlCmd, JRIOBufLengthInvalid.
EIO	One of the following occurred: • The process group of the process that is issuing the function is an orphaned, background process group, and the process that is issuing the function is not ignoring or blocking **SIGTTOU**. • There has been a network or transport failure. The following reason codes can accompany this return code: JRPrevSockError.
ENODEV	The device is incorrect. The function is not supported by the device driver. The following reason code can accompany this return code: JRFuncNotSupported.

Return Code	Explanation
ENOTTY	*file_descriptor* is incorrect. The file type is not character special. The following reason code can accompany this return code: JRNotSupportedForFileType.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

write (BPX1WRT) — Write to a File or Socket

```
BPX1WRT
        file_descriptor
        buffer_address
        buffer_ALET
        write_count
        return_value
        return_code
        reason_code
```

Purpose

Use the write (BPX1WRT) service to write data from a buffer to an open file or socket.

Note: The write service is not related to the `write` shell command.

Parameters

file_descriptor
(input,INT,4) is a variable for specifying the descriptor of the open file or socket. where data is to be written.

buffer_address
(input,INT,4) is a variable for specifying the starting address of a buffer containing the data to be written to the file or socket.

buffer_ALET
(input,INT,4) is a variable for specifying the access list entry token (ALET) for *buffer_address*.

Note: This parameter is ignored.

write_count
(input,INT,4) is a variable for specifying the number of bytes of data to be written to the file.

return_value
(output,INT,4) is a variable where the service returns the actual number of bytes written to the file if the request is successful, or `-1` if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is `-1`.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is `-1`.

Usage Notes

File Offset — If the *file_descriptor* parameter identifies a regular file or any other type of file on which you can seek, the service begins writing at the file offset associated with that file descriptor. A successful write operation increments the file offset by the number of bytes written. If the incremented file offset is greater than the previous length of the file, the file is extended; that is, the length of the file is set to the new file offset.

If the file descriptor refers to a file on which you cannot seek, the service begins writing at the current position. No file offset is associated with such a file.

If the file was opened with the "append" option, the service sets the file offset to the end of the file before writing output.

Number of Bytes Written — Ordinarily, the number of bytes written to the output file is the number you specify in the *write_count* parameter. The value of *write_count* is not checked against any system limit, although a limit can be imposed by a high-level-language POSIX implementation.

If you specify a write count of zero bytes, the service returns a return value of zero without attempting any other action.

If you specify a write count that is greater than the space remaining on the output device, fewer bytes than you requested are written. When at least 1 byte is written, the write is considered successful. The return value shows the number of bytes actually written. An attempt to write again to the same file, however, causes an ENOSPC error unless you are using a terminal. With a terminal, if there is not enough room in the buffer for the whole write, the number of bytes that fit are written and the number of bytes actually written is returned in the return value. However, on the next write attempt (assuming the buffer is still full), the write is blocked or EAGAIN is returned, depending on whether the file was opened blocking or nonblocking.

Similarly, fewer bytes than requested are written if the service is interrupted by a signal after some but not all the specified number of bytes are written. The return value shows the number of bytes written. But if no bytes were written before the routine was interrupted, the return value is -1 and an EINTR error is reported.

SIGTTOU Processing — This service causes signal **SIGTTOU** to be sent if all the following conditions are met:

- The process is attempting to write to its controlling terminal.
- TOSTOP is set as a terminal attribute (see “tcgetattr (BPX1TGA) — Get the Attributes for a Terminal” on page 358 or “tcsetattr (BPX1TSA) — Set the Attributes for a Terminal” on page 365).
- The process is running in a background process group.
- The **SIGTTOU** signal is not blocked or ignored.
- The process is not an orphan.

Characteristics and Restrictions

If the file was opened by an authorized program, all subsequent reads and writes against the file must be issued from an authorized state.

Example

The following code writes 80 bytes from the specified buffer to the file specified (FILEDESC). This example follows the rules of reentrancy. For linkage information, see Appendix D, “Reentrant and Nonreentrant Linkage Examples,” on page 551.

```
*        MVC   FILEDESC,            File descriptor from open
         MVC   BUFLENA,=F'80'
         LA    R15,BUFFERA
         ST    R15,BUFA
         SPACE ,
         CALL  BPX1WRT,             Write to a file                  +
               (FILEDESC,           Input: File descriptor           +
               BUFA,                Input: ->Buffer                  +
               PRIMARYALET,         Input: Buffer ALET               +
               BUFLENA,             Input: Number of bytes to write  +
               RETVAL,              Return value: -1 or bytes written +
               RETCODE,             Return code                      +
               RSNCODE),            Reason code                      +
               VL,MF=(E,PLIST)      ---------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EAGAIN	Blocking is not in effect for the specified file, and output cannot be written immediately.
EBADF	The *file_descriptor* parameter does not contain the descriptor of an open file, or that file is not opened for write services. The following reason codes can accompany this return code: JRFileDesNotInUse, JRFileNotOpen.
ECONNRESET	Connection reset by peer. The following reason code can accompany this return code: JRSocketNotCon.
EFBIG	Writing to the specified file would exceed the maximum file size supported.
EINTR	The service was interrupted by a signal before it could write any data.
EINVAL	The *write_count* parameter contains a value that is less than zero. The following reason code can accompany this return code: JRSocketCallParmError.
EIO	The process is in a background process group and is attempting to write to its controlling terminal. However, TOSTOP is set, the process is neither ignoring nor blocking SIGTTOU signals, and the process group of the process is orphaned. For example, this can happen if a background job tries to write to the terminal after the user has logged off.
EMSGSIZE	The message was too large to be sent all at once, as socket protocol requires.
ENOBUFS	A buffer could not be obtained.
ENOSPC	There is no space left on the output device.
ENOTCONN	The socket was not connected. The following reason code can accompany this return code: JRSocketNotCon.
EPIPE	The request is for a write to a pipe that is not open for reading by any other process. This error also generates a SIGPIPE signal.
EWOULDBLOCK	A write was requested that would have caused a nonblocking socket to block.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “fcntl (BPX1FCT) — Control Open File Descriptors” on page 88
- “lseek (BPX1LSK) — Change the File Offset” on page 154
- “open (BPX1OPN) — Open a File” on page 181
- “pipe (BPX1PIP) — Create an Unnamed Pipe” on page 199
- “read (BPX1RED) — Read from a File or Socket” on page 228
- “socket (BPX1SOC) — Create a Socket” on page 330.

writev (BPX1WRV) — Write Data from a Set of Buffers

```
BPX1WRV
        socket_descriptor
        IOV_count
        IOV_structures
        IOV_ALET
        IOV_buffer_ALET
        return_value
        return_code
        reason_code
```

Purpose

Use the writev (BPX1WRV) service to write data from a set of buffers to a socket.

Parameters

socket_descriptor
(input,INT,4) is a variable for specifying the descriptor of the socket.

IOV_count
(input,INT,4) is a variable for specifying the number of buffers that are pointed to by *IOV_structures*.

IOV_structures
(input,CHAR,*IOV_count* times length of BPXYIOV) is a variable for specifying the IOV structures that contain information about the buffers from which data is to be retrieved. The IOV structure is mapped by the BPXYIOV macro. See "BPXYIOV — Map the I/O Vector Structure" on page 430.

IOV_ALET
(input,INT,4) is a variable for specifying the access list entry token (ALET) for *IOV_structures*.

Note: This parameter is ignored.

IOV_buffer_ALET
(input,INT,4) is a variable for specifying the ALET for the buffers that are pointed to by *IOV_structures*.

Note: This parameter is ignored.

return_value
(output,INT,4) is a variable where the service returns the number of bytes written from the buffers if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

Number of Bytes Written — Ordinarily, the number of bytes written to the output file is the number you request for writing. The number of bytes requested for writing is not checked against any system limit, although a limit can be imposed by a high-level-language POSIX implementation.

If you request zero bytes, the service returns a return value of zero without attempting any other action.

If you request a number of bytes that is greater than the remaining space on the output device, or greater than the file size limit of the physical file system, fewer bytes than you requested are written. When at least 1 byte is written, the write is considered successful. The return value shows the number of bytes actually written. An attempt to write again to the same file, however, causes an error. An error of ENOSPC is returned if there is no remaining space on the output device. An error of EFBIG is returned if the file size limit for the physical file system is exceeded.

Similarly, fewer bytes that requested are written if the service is interrupted by a signal after some but not all of the specified number of bytes are written. The return value shows the number of bytes written. But if no bytes were written before the routine was interrupted, the return value is -1 and an EINTR error is reported.

SIGTTOU Processing — This service causes signal **SIGTTOU** to be sent if all the following conditions are met:

- TOSTOP is set as a terminal attribute (see "tcgetattr (BPX1TGA) — Get the Attributes for a Terminal" on page 358 or "tcsetattr (BPX1TSA) — Set the Attributes for a Terminal" on page 365).
- The process is running in a background process group.
- The **SIGTTOU** signal is not blocked or ignored.
- The process is not an orphan.

Example

The following code issues a writev for a socket. SOCKDESC was returned from a previous call to either socket (BPX1SOC) or accept (BPX1ACP). This example follows the rules of reentrancy. For linkage information, see Appendix D, "Reentrant and Nonreentrant Linkage Examples," on page 551. For the data structures, see "BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services" on page 465 and "BPXYIOV — Map the I/O Vector Structure" on page 430.

```
         MVC   BUFFERA(16),=CL16'Here is the data'
         LA    R2,BUFFERA
         ST    R2,IOV_BASE
         MVI   IOV_LEN,16
*
         CALL  BPX1WRV,              Write from a vector of buffers    +
               (SOCKDESC,            Input: Socket Descriptor          +
               =A(1),                Input: Single element in iov      +
               IOV,                  Input: Iov containing info        +
               PRIMARYALET,          Input: Alet where iov resides     +
               PRIMARYALET,          Input: Alet of buffers for data   +
               RETVAL,               Return value: Num bytes or -1     +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EBADF	An incorrect file descriptor was specified. The following reason codes can accompany this return code: JRFileDesNotInUse, JRFileNotOpen, JRRFileWrOnly, JRWFileRdOnly.
ECONNRESET	Connection reset by peer. The following reason code can accompany this return code: JRSocketNotCon.
EINTR	A signal interrupted the writev service before any data was written.
EINVAL	One of the input parameters was incorrect. The following reason codes can accompany this return code: JRBytes2RWZero, JROutofRange, JRSocketCallParmError.

Return Code	Explanation
EIO	The process is in a background process group and is attempting to write to its controlling terminal. However, TOSTOP is set, the process is neither ignoring nor blocking **SIGTTOU** signals, and the process group of the process is orphaned. This can happen, for example, if a background job tries to write to the terminal after the user has logged off.
EMSGSIZE	The message is too large to be sent all at once, as the socket requires.
ENOBUFS	A buffer could not be obtained.
ENOTCONN	The socket was not connected. The following reason code can accompany this return code: JRSocketNotCon.
ENOTSOCK	*socket_descriptor* does not refer to a valid socket descriptor. The following reason code can accompany this return code: JRMustBeSocket.
EPIPE	An attempt was made to write to a socket that is shut down or closed. This error also generates a **SIGPIPE** signal.
EPROTOTYPE	An incorrect socket type was supplied. The following reason code can accompany this return code: JRIncorrectSocketType.
EWOULDBLOCK	A write was requested that would have caused a nonblocking socket to block.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Service

Another callable service related to this service is:

- “readv (BPX1RDV) — Read Data and Store It in a Set of Buffers” on page 238

w_statvfs (BPX1STF) — Get Status Information about a File System by File System Name

BPX1STF

file_system_name
status_area_length
status_area
return_value
status_area
return_code
reason_code

Purpose

Use the w_statvfs (BPX1STF) service to obtain status information about a file system by its file system name.

For the corresponding service using a file descriptor, see "fstatvfs (BPX1FTV) — Get Status Information about File System by Descriptor" on page 104. For the corresponding service using a path name, see "statvfs (BPX1STV) — Get Status Information about a File System by Path Name" on page 343.

Parameters

file_system_name
(input,INT,4) is a variable for specifying the file system name whose status is to be returned.

status_area_length
(input,INT,4) is a variable for specifying the length of the *status_area* parameter.

status_area
(output,CHAR,*status_area_length*) is a variable for the area where the service returns the status information for the file system. This area is mapped by the BPXYSSTF macro. See "BPXYSSTF — Map the File System Status Structure" on page 471.

return_value
(output,INT,4) is a variable where the service returns the length of the data returned in *status_area* if the request is successful, or -1 if it is not successful.

return_code
(output,INT,4) is a variable where the service stores the return code. A return code is returned only if *return_value* is -1.

reason_code
(output,INT,4) is a variable where the service stores the reason code. A reason code is returned only if *return_value* is -1.

Usage Notes

1. It is not considered an error if the passed *status_area_length* is not sufficient to hold all the returned information. (In other words, future expansion is allowed for.) As much information as will fit is written to *status_area*, and this amount is returned.
2. If a buffer of length of zero is passed to this service, no data is returned and the return value is zero. You can check for the existence of a file system by passing such a length.
3. The amount of valid data returned in the *status_area* is indicated by the *return_value*. This allows for differences in the release levels of VM and the physical file systems.

Example

The following code requests information about file system TESTLIB.FILESYS1.

```
MVC   FSNAME(44),=CL44'TESTLIB.FILESYS1'
  SPACE ,
  CALL  BPX1STF,              Get file system status            +
        (FSNAME,              Input: File system name (44 char) +
        SSTFL,                Input: Length of BPXYSSTF         +
        SSTF,                 Buffer, BPXYSSTF                  +
        RETVAL,               Return value: Status length or -1 +
        RETCODE,              Return code                       +
        RSNCODE),             Reason code                       +
        VL,MF=(E,PLIST)       ----------------------------------
```

Return Codes and Reason Codes

This service can return the following return codes:

Return Code	Explanation
EAGAIN	Information is temporarily unavailable. This can occur because the mount process for the file system is incomplete.
EINVAL	Parameter error. For example, *file_system_name* was not found. The following reason code can accompany this return code: JRFileSysNotThere.

For a complete list of return codes for OpenExtensions callable services, see Appendix A, “Return Codes,” on page 487. For a complete list of reason codes for OpenExtensions callable services, with explanations and required actions, see Appendix B, “Reason Codes,” on page 495.

Related Services

Other callable services related to this service are:

- “fstatvfs (BPX1FTV) — Get Status Information about File System by Descriptor” on page 104.
- “statvfs (BPX1STV) — Get Status Information about a File System by Path Name” on page 343.

Chapter 3. Mapping Macro Descriptions

The Mapping macros described in this chapter map the parameter options, constants, and data returned in many OpenExtensions callable services. If a macro field contains the comment "Reserved for IBM Use" or similar words, that field is not a programming interface for customer use.

Most of the mapping macros can be expanded with or without a DSECT statement. The invocation parameter DSECT=YES (the default) can be used with either reentrant or nonreentrant programs with the appropriate rules governing the storage backed by the USING statement.

Many of the mapping macros exploit the fact that DC expands as a DS in a DSECT and as a DC with its initialized value in a CSECT. When these fields are expanded as or within DSECTs, the program is responsible for initializing the necessary fields.

To assemble a program using any of these macros, you must issue the GLOBAL command specifying MACLIB DMSGPI. This macro library is usually located on the system disk.

Understanding the Macro Syntax Diagrams

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

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

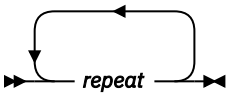
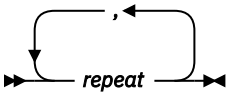
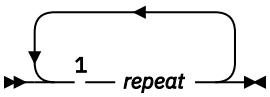
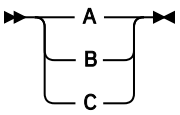
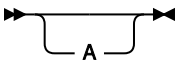
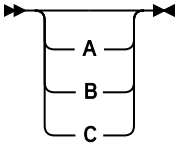
- The ►►—— symbol indicates the beginning of a syntax diagram.
- The ——► symbol, at the end of a line, indicates that the syntax diagram continues on the next line.
- The ►—— symbol, at the beginning of a line, indicates that a syntax diagram continues from the previous line.
- The ——►◄ symbol indicates the end of a syntax diagram.

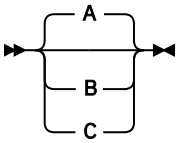
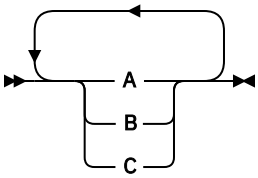
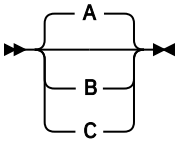
Syntax items (for example, a keyword or variable) may be:

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

Syntax Diagram Description	Example
Abbreviations: Uppercase letters denote the shortest acceptable abbreviation. If an item appears entirely in uppercase letters, it cannot be abbreviated. You can type the item in uppercase letters, lowercase letters, or any combination. In this example, you can enter KEYWO, KEYWOR, or KEYWORD in any combination of uppercase and lowercase letters.	►► KEYWOrd ►◄

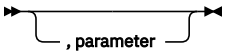
© Copyright IBM Corp. 1993, 2022

Syntax Diagram Description	Example
Symbols: You must code these symbols exactly as they appear in the syntax diagram.	***** Asterisk **:** Colon **,** Comma **=** Equal Sign **-** Hyphen **()** Parentheses **.** Period
Variables: Highlighted lowercase items (*like this*) denote variables. In this example, *var_name* represents a variable you must specify when you code the KEYWORD command.	
Repetition: An arrow returning to the left means that the item can be repeated. A character within the arrow means you must separate repeated items with that character. A footnote (1) by the arrow references a limit that tells how many times the item can be repeated.	   Notes: [1] Specify *repeat* up to 5 times.
Required Choices: When two or more items are in a stack and one of them is on the line, you *must* specify one item. In this example, you must choose A, B, or C.	
Optional Choice: When an item is below the line, the item is optional. In this example, you can choose A or nothing at all. When two or more items are in a stack below the line, all of them are optional. In this example, you can choose A, B, C, or nothing at all.	 

Syntax Diagram Description	Example
Defaults: Defaults are above the line. The system uses the default unless you override it. You can override the default by coding an option from the stack below the line. In this example, A is the default. You can override A by choosing B or C.	
Repeatable Choices: A stack of items followed by an arrow returning to the left means that you can select more than one item or, in some cases, repeat a single item. In this example, you can choose any combination of A, B, or C.	
Syntax Fragments: Some diagrams, because of their length, must fragment the syntax. The fragment name appears between vertical bars in the diagram. The expanded fragment appears in the diagram after a heading with the same fragment name. In this example, the fragment is named "A Fragment."	 **A Fragment** 

Coding Conventions

Coding conventions for OpenExtensions macros are the same as those for all assembler language macros. The macro format descriptions show optional parameters in the format:



indicating that if you are going to use this parameter, it must be preceded by a comma (unless it is the first parameter coded). If a macro statement overflows to a second line, you must use a continuation character in column 72.

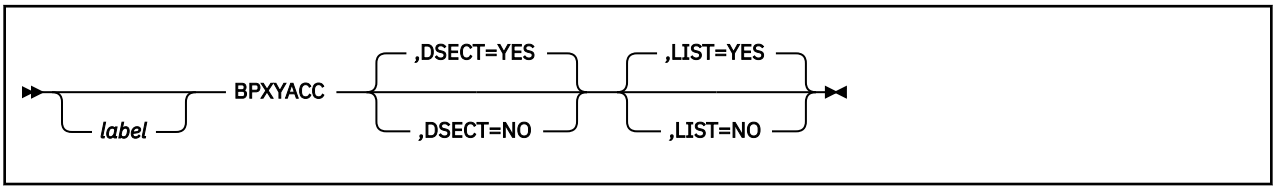
Note: No blanks may appear between parameters.

When a macro offers a choice of parameters, one and only one of which must be specified, the parameters are stacked one per line and shown below the line of the syntax diagram.

Many operands can be specified with an argument in the form of either an expression or a register containing a value. When this is the case, the macro expects a register designation to begin with a left parenthesis. Therefore, specifying an expression that starts with a left parenthesis will produce unpredictable results, just as specifying a register without parentheses would.

Incorrect coding of any macro may result in assembler errors and MNOTEs. MNOTES are unnumbered responses that can result from executing system generation macroinstructions or service programs. They are documented in logic listings only.

BPXYACC — Map Flag Values for the access Service



Purpose

Use the BPXYACC macro to map flag values for the access (BPX1ACC) callable service.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

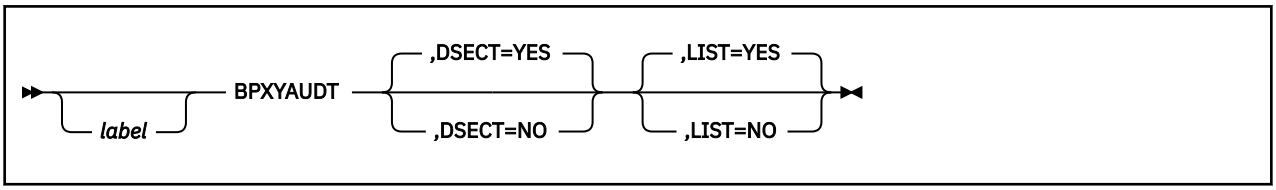
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYACC mapping macro expands as follows:

```
         BPXYACC    ,
ACC                 DSECT ,
ACCRSRV             DS    CL3    Reserved for IBM use
ACCINTENTFLAGS      DS    XL1    Access Intent Flags
*                   EQU   X'F0'  Reserved for IBM use
ACC_F_OK            EQU   X'08'  Check for file existence
ACC_R_OK            EQU   X'04'  Check for read access to file
ACC_W_OK            EQU   X'02'  Check for write access to file
ACC_X_OK            EQU   X'01'  Check for execute access to file
ACC#LENGTH          EQU   *-ACC  Length of this structure
```

BPXYAUDT — Map Flag Values for the chaudit and fchaudit Services



Purpose

Use the BPXYAUDT macro to map flag values for the chaudit (BPX1CHA) and fchaudit (BPX1FCA) callable services.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

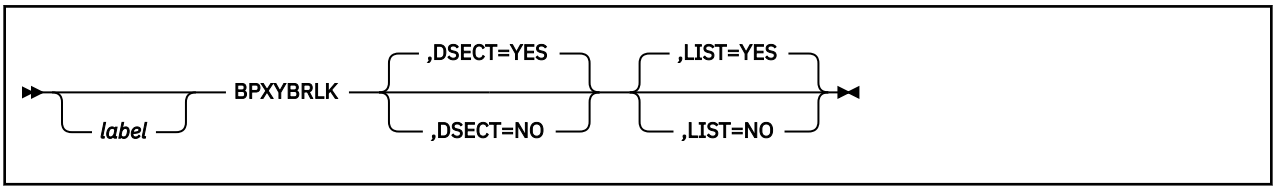
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYAUDT mapping macro expands as follows:

```
         BPXYAUDT   ,
AUDT                DSECT ,
AUDTREADACCESS      DS    XL1    Read Access Auditing Flags
AUDTREADFAIL        EQU   X'02'  1 = audit failing read accesses
AUDTREADSUCC        EQU   X'01'  1 = audit successful read accesses
AUDTWRITEACCESS     DS    XL1    Write Access Auditing Flags
AUDTWRITEFAIL       EQU   X'02'  1 = audit failing write accesses
AUDTWRITESUCC       EQU   X'01'  1 = audit successful write accesses
AUDTEXECACCESS      DS    XL1    Execute/Search Auditing Flags
AUDTEXECFAIL        EQU   X'02'  1 = audit failing exec or search
AUDTEXECSUCC        EQU   X'01'  1 = audit successful exec or search
AUDTRSRV            DS    XL1    Flag byte 4 -Reserved for IBM use
AUDT#LENGTH         EQU   *-AUDT Length of this structure
```

BPXYBRLK — Map the Byte Range Lock Request for the fcntl Service



Purpose

Use the BPXYBRLK macro to map the byte range lock request for the fcntl (BPX1FCT) callable service.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

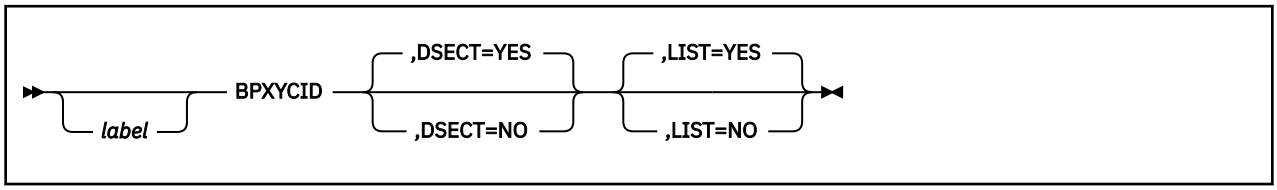
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYBRLK mapping macro expands as follows:

```
         BPXYBRLK   ,
BRLK                DSECT ,
L_TYPE              DS    H      Requested lock type:
F_RDLCK             EQU   1       Shared or read lock
F_WRLCK             EQU   2       Exclusive or write lock
F_UNLCK             EQU   3       Unlock
L_WHENCE            DS    H      Flag for starting offset
L_START             DS    0CL8   Relative offset in bytes
L_START_H           DS    F      High word of relative offset
L_START_L           DS    F      Low word of relative offset
L_LEN               DS    0CL8   Size of lock in bytes
L_LEN_H             DS    F      High word of size of lock in bytes
L_LEN_L             DS    F      Low word of size of lock in bytes
L_PID               DS    F      Process ID of process holding lock
BRLK#LENGTH         EQU   *-BRLK Length of this area
```

BPXYCID — Map the Client ID Structure



Purpose

Use the BPXYCID macro to map the client ID data structure returned by the getclientid (BPX1GCL) callable service and used by the givesocket (BPX1GIV) and takesocket (BPX1TAK) callable services.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

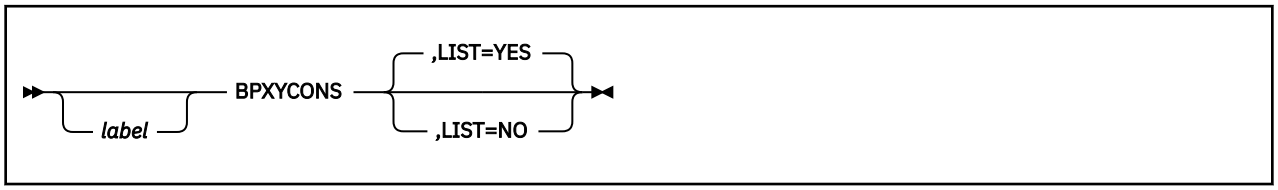
Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYCID macro expands as follows:

```
            BPXYCID    ,
*
CID                 DSECT ,          ClientId structure
CIDBEGIN            DS    0D
*
CIDDOMAIN           DS    F          Domain
CIDNAME             DS    CL8        Address space name
CIDTASK             DS    CL8        Subtask name
CIDRESERVED         DS    CL20       Reserved
*
CID#LENGTH          EQU   *-CID      Constant - Fixed length of CID
*
                    ORG   CIDNAME
CIDNAMEUPPER        DS    F          Binary zeroes
CIDPID              DS    F          Process Id
*
                    ORG   CIDRESERVED
CIDTYPE             DS    X          Type of request
CIDSPECIFIC         DS    CL19
*
                    ORG   CIDSPECIFIC
                    DS    CL3
CIDSOCKTOKEN        DS    F          Returned token
                    ORG   ,
*
CID#CLOSE           EQU   1          Close socket
CID#SELECT          EQU   2          Giver will do select
```

```
*
*
*
** BPXYCID End
```

BPXYCONS — Map Constants



Purpose

Use the BPXYCONS macro to map the constants used by OpenExtensions callable services.

Parameters

label
is an optional assembler label for the statement.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

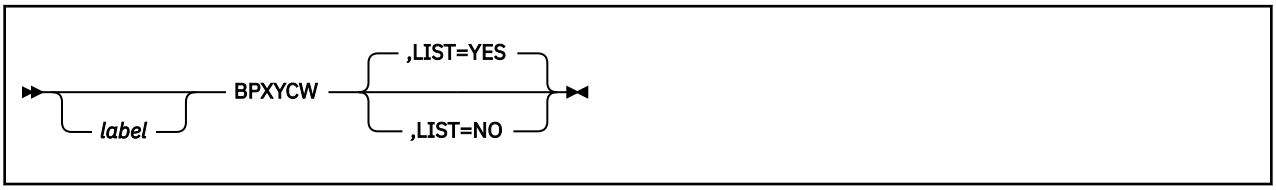
Usage Notes

1. The DSECT= parameter is allowed but ignored.
2. The PRINT OFF assembler statement overrides LIST=YES.
3. The BPXYCONS macro expands as follows:

```
         BPXYCONS   ,
DFLT_ARG_MAX         EQU   1048576 Constant for default ARG_MAX (1 MEG)
DFLT_CHILD_MAX       EQU   32   Constant for default CHILD_MAX
*
DFLT_CLK_TCK         EQU   100  Constant for default CLK_TCK
*                              (100 ticks per second)
DFLT_OPEN_MAX        EQU   65536 Constant for default OPEN_MAX
*                              (_POSIX_OPEN_MAX)
DFLT_TZNAME_MAX      EQU   5    Constant for default TZNAME_MAX
DFLT_JOB_CONTROL     EQU   1    Constant for default JOB_CONTROL
DFLT_SAVED_IDS       EQU   1    Constant for default SAVED_IDS
DFLT_VERSION         EQU   199009  Constant for default VERSION
DFLT_THREAD_TASKS_MAX_NP  EQU  -1  Constant default THREAD_TASKS_MAX_NP
DFLT_2_CHAR_TERM     EQU  -1 Constant default THREAD_TASKS_MAX_NP
* items from sysconf()
SC_ARG_MAX           EQU   1    Constant for querying ARG_MAX
SC_CHILD_MAX         EQU   2    Constant for querying CHILD_MAX
SC_CLK_TCK           EQU   3    Constant for querying CLK_TCK
SC_JOB_CONTROL       EQU   4    Constant for querying JOB_CONTROL
SC_NGROUPS_MAX       EQU   5    Constant for querying NGROUPS_MAX
SC_OPEN_MAX          EQU   6    Constant for querying OPEN_MAX
SC_SAVED_IDS         EQU   7    Constant for querying SAVED_IDS
SC_TZNAME_MAX        EQU   9    Constant for querying TZNAME_MAX
SC_VERSION           EQU   10   Constant for querying VERSION
SC_THREAD_TASKS_MAX_NP EQU 11   Constant to query THREAD_TASKS_MAX_NP
SC_2_CHAR_TERM       EQU   12   Constant for querying VERSION
* wait function code
#WAITID              EQU   2    waitid() function code
* items from wait()
WNOHANG              EQU   1    Wait, do not suspend execution
WUNTRACED            EQU   2    Wait, return status of stopped child
WCONTINUED           EQU   4    Wait, return status of continued child
WEXITED              EQU   8    Wait for processes that have exited
WSTOPPED             EQU  16    Wait, return status of stopped child
WNOWAIT              EQU  32    Wait, return status of a child without
*                               changing the state.  The child can be
*                               waited for again.
* waitid() id type options
```

```
P_PID                EQU   0    Wait for the child with a process ID
P_PGID               EQU   1    Wait for any child with a process
*                               group ID
P_ALL                EQU   2    Wait for any child
*
SPAWN_FDCLOSED       EQU  -1    Do not inherit this file descriptor
PTEXITTHREAD         EQU   0    Pthread exit
PTGETNEWTHREAD       EQU   1    Pthread get new
PTFAILIFLASTTHREAD   EQU   2    Pthread fail if last thread
QUIESCE_TERM         EQU   1    quiesce_threads type = term
QUIESCE_FORCE        EQU   2    quiesce_threads type = force
QUIESCE_QUERY        EQU   3    Alias of pthread_query
PTHREAD_QUERY        EQU   3    quiesce_threads type = query
PTHREAD_INTR_ENABLE#  EQU  0    Cancel request type = enabled
PTHREAD_INTR_DISABLE# EQU  1    Cancel request type = disabled
PTHREAD_INTR_CONTROLLED#   EQU 0  Cancel request type = controlled
PTHREAD_INTR_ASYNCHRONOUS# EQU 1  Cancel request type = Asynchronous
STDIN_FILENO         EQU   0    Standard input value, file descriptor
STDOUT_FILENO        EQU   1    Standard output value, file descriptor
STDERR_FILENO        EQU   2    Standard error value, file descriptor
*   The high-order two bytes of the reason codes returned by
*   OpenExtensions services contains a value that is used to
*   qualify the contents of the low order two bytes. If the contents
*   of the high-order two bytes are within the range of #CMID_LO to
*   #CMID_HI, the error represented by the reason code is defined
*   by OpenExtensions. If the contents of the high-order two bytes
*   lie outside the range, the error represented by the reason code
*   is not an OpenExtensions reason code.
#CMID_LOW            EQU   0000   Low range
#CMID_HI             EQU   8447   High range
```

BPXYCW — Map Serialization Constants



Purpose

Use the BPXYCW macro to map the serialization constants used by OpenExtensions callable services.

Parameters

label
is an optional assembler label for the statement.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

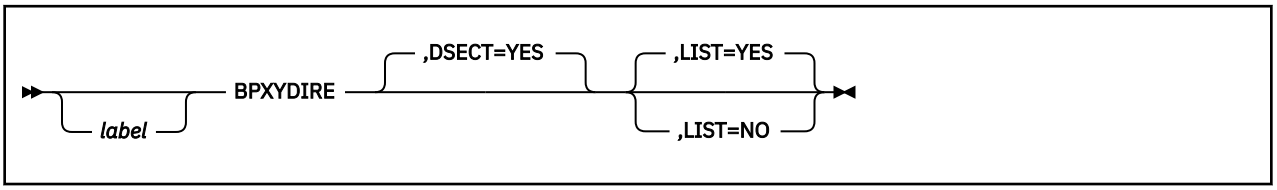
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The DSECT= parameter is allowed but ignored.
2. The PRINT OFF assembler statement overrides LIST=YES.
3. The BPXYCW macro expands as follows:

```
         BPXYCW   ,
CW_INTRPT        EQU   1      Thread interrupted by a signal
*                                                (x'0000 0001')
CW_CONDVAR       EQU   32     Thread notified that some condition
*                               has been met     (x'0000 0020')
CW_TIMEOUT       EQU   64     Timeout occurred   (x'0000 0040')
*
```

BPXYDIRE — Map Directory Entries for the readdir Service



Purpose

Use the BPXYDIRE macro to map directory entries for the readdir (BPX1RDD) callable service.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

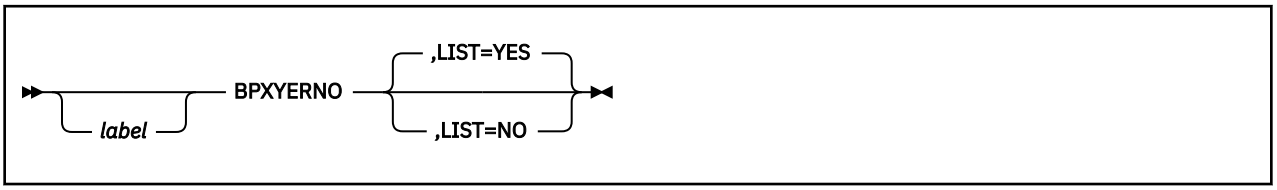
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. DSECT=NO is not allowed. The basing for the PFSOTHER data is not known, as it depends on the length of the name.
2. The PRINT OFF assembler statement overrides LIST=YES.
3. The BPXYDIRE mapping macro expands as follows:

```
         BPXYDIRE   ,
* LA     RegOne,buffer            RegOne->BPX1RDD buffer and 1st DIRE
* USING  DIRE,RegOne              Addressability to DIRE
DIRE                DSECT ,
DIRENTINFO          DS    0X     Fixed length information
DIRENTLEN           DS    H      Entry length
DIRENTNAML          DS    H      Name length
DIRENTNAME          DS    0C     Name
* LR     RegTwo,RegOne            RegTwo->DIRE
* A      RegTwo,=F'4'             RegTwo->start of name
* A      RegTwo,DIRENTNAML        RegTwo->end of name+1
* USING  DIRENTPFSDATA,RegTwo     Addressability to DIRENTPFSDATA
DIRENTPFSDATA       DSECT ,      Physical file system-specific data
DIRENTPFSINO        DS    CL4    File Serial Number = st_ino
DIRENTPFSOTHER      DS    0C     Other PFS specific data
* A      RegOne,DIRENTLEN         RegOne->Next DIRE in buffer
* BCT    Return_Value,Back_to_process_next_DIRE
```

BPXYERNO — Map Return Codes and Reason Codes



Purpose

Use the BPXYERNO macro to map the values for the return codes and reason codes generated by OpenExtensions callable services. BPXYERNO consists only of equates.

Parameters

label
is an optional assembler label for the statement.

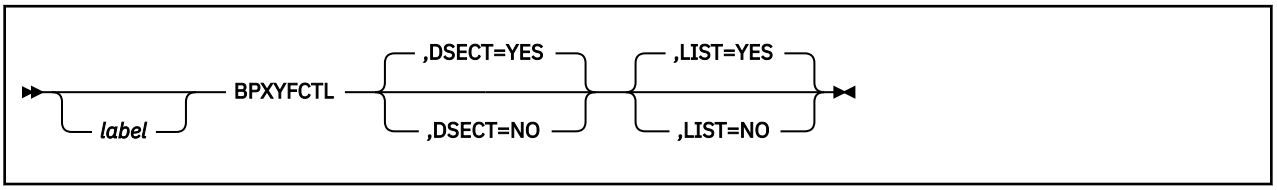
LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The DSECT= parameter is allowed but ignored.
2. The PRINT OFF assembler statement overrides LIST=YES.
3. For information on the return codes and reason codes defined in the BPXYERNO macro, see Appendix A, “Return Codes,” on page 487 and Appendix B, “Reason Codes,” on page 495.

BPXYFCTL — Map Command Values and Flags for the fcntl Service



Purpose

Use the BPXYFCTL macro to map command values and flags for the fcntl (BPX1FCT) callable service.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

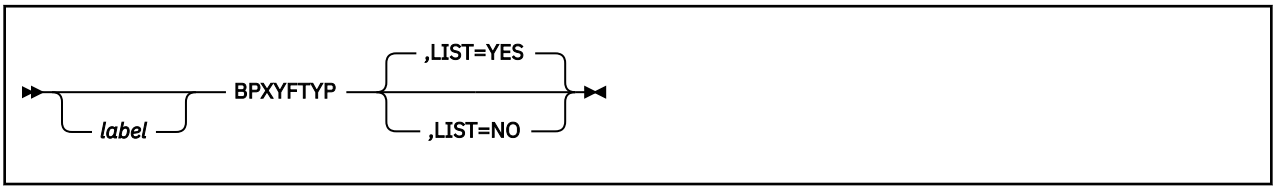
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYFCTL mapping macro expands as follows:

```
         BPXYFCTL   ,
FCTL                DSECT ,
*                                External file descriptor flags
FCTLFDFL1           DS    B
FCTLRS01            EQU   X'80'  Reserved for IBM use
*                                FCTLFDFLAGS must never be < 0
FCTLFDFL2           DS    B      Reserved for IBM use
FCTLFDFL3           DS    B      Reserved for IBM use
FCTLFDFL4           DS    B
FCTLCLOFORK         EQU   X'02'  1= close_on_fork
FCTLCLOEXEC         EQU   X'01'  1= close_on_exec
*                                Command value definitions
F_DUPFD             EQU   0      Duplicate file descriptor
F_GETFD             EQU   1      Get file descriptor flags
F_SETFD             EQU   2      Set file descriptor flags
F_GETFL             EQU   3      Set file status flags
F_SETFL             EQU   4      Set file status flags
F_GETLK             EQU   5      Get record locking information
F_SETLK             EQU   6      Set record locking information
F_SETLKW            EQU   7      Set record locking information,
*                                wait if blocked
F_DUPFD2            EQU   8      Duplicate file descriptor, option 2
F_CLOSFD            EQU   9      Close file descriptors
FCTL#LENGTH         EQU   *-FCTL Length of this structure
** BPXYFCTL End
```

BPXYFTYP — Map File Type Definitions



Purpose

Use the BPXYFTYP macro to map the file type definitions for OpenExtensions callable services.

Parameters

label
is an optional assembler label for the statement.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

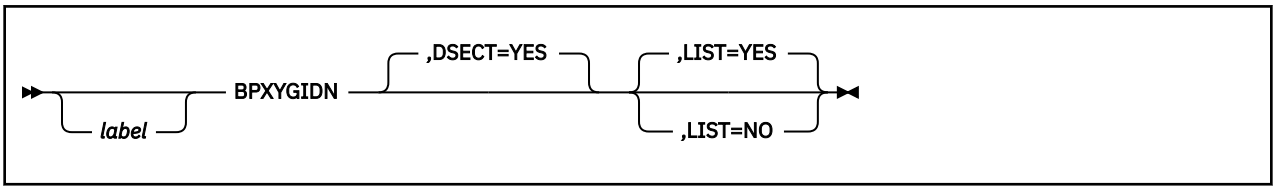
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The DSECT= parameter is allowed but ignored.
2. The PRINT OFF assembler statement overrides LIST=YES.
3. The BPXYFTYP mapping macro expands as follows:

```
         BPXYFTYP   ,
FT_DIR               EQU   1      Directory File
FT_CHARSPEC          EQU   2      Character Special File
FT_REGFILE           EQU   3      Regular File
FT_FIFO              EQU   4      Named Pipe (FIFO) File
FT_SYMLINK           EQU   5      Symbolic link
FT_BLKSPEC           EQU   6      Reserved for Block Special
FT_SOCKET            EQU   7      Sockets
FT_EXTLINK           EQU   254    External Link
**
** External Link Subtypes
FST_EXEC             EQU   1      Executable
FST_DATA             EQU   2      Data
FST_MEL              EQU   3      Mount
FST_SOCKET           EQU   4      Socket
```

BPXYGIDN — Map the Data Structure Returned for the getpwnam and getpwuid Services



Purpose

Use the BPXYGIDN macro to map the data structure returned for the getpwnam (BPX1GPN) and getpwuid (BPX1GPU) callable services.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

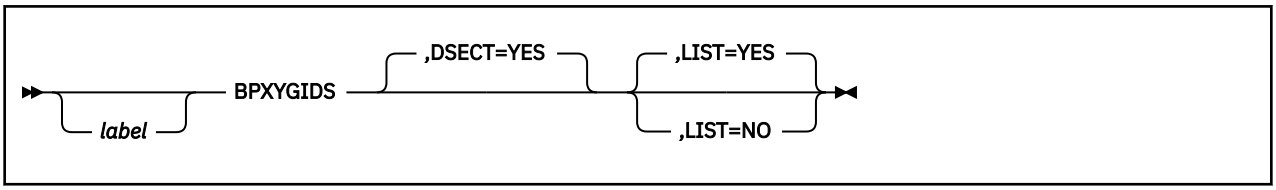
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. DSECT=NO is not allowed. The storage belongs to the service and a pointer is returned to the invoker.
2. The PRINT OFF assembler statement overrides LIST=YES.
3. The BPXYGIDN mapping macro expands as follows:

```
         BPXYGIDN   ,
GIDN                 DSECT ,        USING on Return_value of GPN
GIDN_U_LEN           DS    F        Length of GIDN_U_NAME
GIDN_U_NAME          DS    0C       User name
*  Add GIDN_U_LEN to Index or base to access next field
                     DS    F        Length of user ID               4
GIDN_USERID          DS    F        User ID
                     DS    F        Length of group ID              4
GIDN_GROUPID         DS    F        Group ID
GIDN_D_LEN           DS    F        Length of GIDN_D_NAME       0-1023
GIDN_D_NAME          DS    0C       Initial working directory name
*  Add GIDN_D_LEN to Index or base to access next field
GIDN_P_LEN           DS    F        Length of GIDN_P_NAME       0-1023
GIDN_P_NAME          DS    0C       Initial user program name
GIDN_F_LEN           DS    F        Length of GIDN_F_NAME       0-1023
GIDN_F_NAME          DS    0C       FSROOT pathid
GIDN#LENGTH          EQU   *-GIDN   Length less U_LEN, D_LEN and P_LEN
```

BPXYGIDS — Map the Data Structure Returned for the getgrnam and getgrgid Services



Purpose

Use the BPXYGIDS macro to map the data structure returned for the getgrnam (BPX1GGN) and getgrgid (BPX1GGI) callable services.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

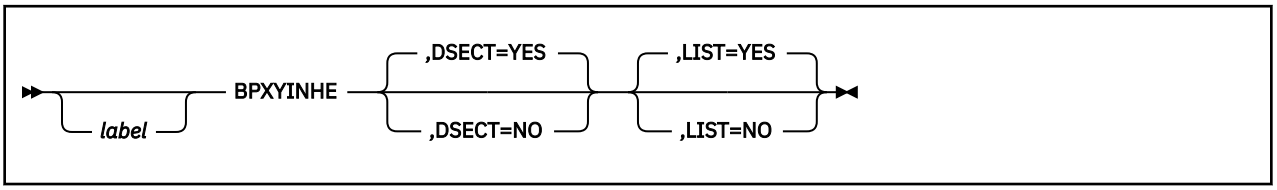
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. DSECT=NO is not allowed. The storage belongs to the service and a pointer is returned to the invoker.
2. The PRINT OFF assembler statement overrides LIST=YES.
3. The BPXYGIDS mapping macro expands as follows:

```
         BPXYGIDS   ,
GIDS                 DSECT ,
GIDS_G_LEN           DS    F        Length of GIDS_G_NAME
GIDS_G_NAME          DS    0C       Group name
* Add GIDS_G_LEN to index or base to access following fields
                     DS    F        Length of group ID, always 4
GIDS_GROUPID         DS    F        Group ID
GIDS_COUNT           DS    F        Count of array elements
* Make a local copy of GIDS_COUNT
* Test: if local copy of GIDS_COUNT zero, quit
GIDS_M_LEN           DS    F        Length of GIDS_M_NAME
GIDS_M_NAME          DS    0C       Member name
* Add GIDS_M_LEN+4 to index or base
* Decrement local copy of GIDS_COUNT, goto test.
GIDS#LENGTH          EQU   *-GIDS   Length less all variable fields
```

BPXYINHE — Map the Inheritance Structure for the spawn Service



Purpose

Use the BPXYINHE macro to map the inheritance structure used by the spawn (BPX1SPN) service.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

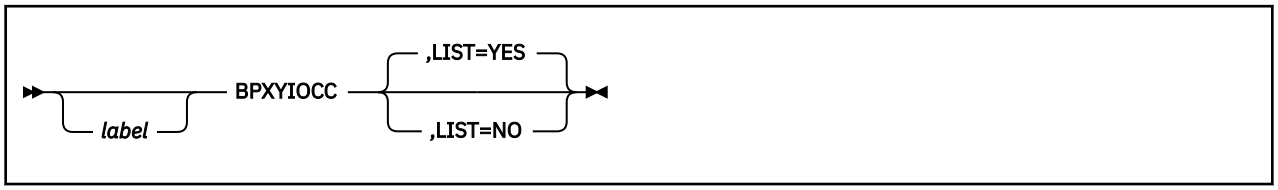
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYINHE macro expands as follows:

```
          BPXYINHE   ,
** BPXYINHE: Inheritance Area
**  Used By: spawn() callable service
INHE                DSECT ,
INHEBEGIN           DS    0D
*
INHEEYE             DC    C'INHE'  Eye catcher
INHELENGTH          DC    AL2(INHE#LENGTH)
                                Length of this structure
INHEVERSION         DC    AL2(INHE#VER)
INHE#VER            EQU   1     Version of this structure
INHEFLAGS           DS    0BL4  Flags indicating contents of structure
INHEFLAGS0          DS    XL1   1st byte
INHESETPGROUP       EQU   X'80' Set Process Group using INHEPGROUP
INHESETSIGMASK      EQU   X'40' Set Signal Mask using INHESIGMASK
INHESETSIGDEF       EQU   X'20' Set Signal Defaults using INHESIGDEF
INHESETTCPGRP       EQU   X'10' Set Cntl TTY Pgrp using INHECTLTTYFD
INHEFLAGS1          DS    XL1   2nd byte
INHEFLAGS2          DS    XL1   3rd byte
INHEFLAGS3          DS    XL1   4th byte
INHEPGROUP          DS    F     Process Group for child
INHE#NEWPGROUP      EQU   0     Put child in a new proc grp of its own
INHESIGMASK         DS    BL8   Signal Mask for child
INHESIGDEF          DS    BL8   Set of default signals for child
INHECTLTTYFD        DS    F     Cntl TTY FD for tcsetgrp() in child
INHE#LENGTH         EQU   *-INHE
** BPXYINHE End
```

BPXYIOCC — Map Command Constants for the w_ioctl Service



Purpose

Use the BPXYIOCC macro to map command constants for the w_ioctl (BPX1IOC) callable service. BPXYIOCC consists only of equates.

Parameters

label
is an optional assembler label for the statement.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

Usage Notes

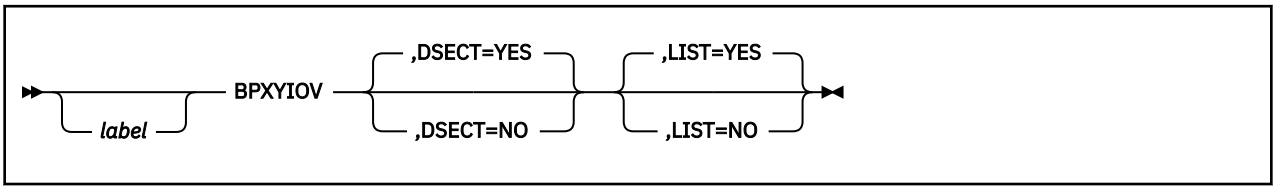
1. The DSECT= parameter is allowed but ignored.
2. The PRINT OFF assembler statement overrides LIST=YES.
3. The BPXYIOCC macro expands as follows:

```
            BPXYIOCC   ,
** BPXYIOCC: Ioctl Command Constant Definitions
**  Used By: Ioctl syscalls
*   Ioctl command constants - Range 1-255 reserved for VM
*  Authorized/Tcpip CMD values
IOCC#TCI              EQU   5000 Cmd for Tcpip Initialization
IOCC#TCC              EQU   5001 Cmd for Complete Tcpip Initialization
IOCC#TCS              EQU   5002 Cmd for Tcpip Path Sever
IOCC#TCR              EQU   5003 Cmd for Tcpip Reply/Post call
IOCC#TCG              EQU   5004 Cmd for Tcpip SiGnal call
IOCC#TCCE             EQU   5006 Cmd for Tcpip End Registration
SIOCMSDELRT           EQU   5007 Cmd for Delete Route
*                                  (Pre-Router wrap)
SIOCMSADDRT           EQU   5008 Cmd for Add Route
*                                  (Pre-Router wrap)
SIOCMSSIFADDR         EQU   5009 Cmd for Set Interface address
*                                  (Pre-Router wrap)
SIOCMSSIFFLAGS        EQU   5010 Cmd for Set Interface Flags
*                                  (Pre-Router wrap)
SIOCMSSIFDSTADDR      EQU   5011 Cmd for Set point-to-point interface
*                                 address (Pre-Router wrap)
SIOCMSSIFBRDADDR      EQU   5012 Cmd for Set Broadcast address
*                                  (Pre-Router wrap)
SIOCMSSIFNETMASK      EQU   5013 Cmd for Set interface network
*                                mask for an Internet address
*                                  (Pre-Router wrap)
SIOCMSSIFMETRIC       EQU   5014 Cmd for Set Interface routing metric
*                                  (Pre-Router wrap)
SIOCMSRBRTTABLE       EQU   5015 Cmd for Set Routing table required
*                                  required request
*                                  (Pre-Router wrap)
SIOMSMETRIC1RT        EQU   5016 Cmd for Set metric1
*                                  (Pre-Router wrap)
SIOCMSICMPREDIRECT    EQU   5017 Cmd for Propagating ICMP redirects
*                                  (Pre-Router wrap)
SIOCSETTKN            EQU   X'8008139A' 5018 Set Tcp/Ip master Tkn
*
```

```
*  Ioctl Command Constants - terminal control
TIOCGWINSZ           EQU   X'4008A368'  get window size
TIOCSWINSZ           EQU   X'8008A367'  set window size
TIOCNOTIFY           EQU   X'8001A364'  notify master by packet
*  Constants for argument when TIOCNOTIFY is specified
IOCC#PWBEGIN         EQU   1            Begin secure data
IOCC#PWEND           EQU   2            End   secure data
*
* Ioctl command constants - for Router query
SIOCGRTTABLE         EQU   X'C008C980'  Gets Network Routing
*                                         Table
SIOCSETRTTD          EQU   X'8008C981'  Set Socket to be attached to
*                                         1 TD
*
FIONBIO              EQU   X'8004A77E'  set/reset nonblock I/O
FIONREAD             EQU   X'4004A77F'  get number of readable bytes
*                                      available
FIOASYNC             EQU   X'8004A77D'  set/clear async I/O
FIOSETOWN            EQU   X'8004A77C'  set owner
FIOGETOWN            EQU   X'4004A77B'  get owner
SECIGET              EQU   X'4010E401'  get security information
SIOCADDRT            EQU   X'8030A70A'  IBM use only, Add routing
*                                      table entry
SIOCATMARK           EQU   X'4004A707'  Is current location pointing
*                                      to out-of-band data?
SIOCDELRT            EQU   X'8030A70B'  IBM use only, Delete routing
*                                      table entry
SIOMETRIC1RT         EQU   X'8030A70C'  IBM use only, Set metric1
SIOCSIFADDR          EQU   X'8020A70C'  Set Network interface addr
SIOCGIFADDR          EQU   X'C020A70D'  Get Network interface address
SIOCGIFBRDADDR       EQU   X'C020A712'  Get Network interface
*                                      Broadcast Address
SIOCSIFBRDADDR       EQU   X'8020A713'  Sets Network interface
*                                      Broadcast Address
SIOCGIFCONF          EQU   X'C008A714'  Get Network interface
*                                      Configuration
SIOCGIFDSTADDR       EQU   X'C020A70F'  Get Network interface
*                                      Destination Address
SIOCGIFFLAGS         EQU   X'C020A711'  Get Network interface Flags
SIOCGIFMETRIC        EQU   X'C020A717'  IBM use only, Gets Network
*                                      Interface Routing Metric
SIOCGIFNETMASK       EQU   X'C020A715'  Get Network interface
*                                      Network Mask
SIOCSIFNETMASK       EQU   X'8020A716'  Set Network interface
*                                      Network Mask
SIOCSIFDSTADDR       EQU   X'8020A70E'  IBM use only, Sets Network
*                                      Interface Destination Address
SIOCSIFFLAGS         EQU   X'8020A710'  IBM use only, Sets Network
*                                      Interface Flags
SIOCSIFMETRIC        EQU   X'8020A718'  IBM use only, Sets Network
*                                      Interface Routing Metric
SIOCSARP             EQU   X'8024A71E'  IBM use only, Sets ARP
*                                      Entry
SIOCGARP             EQU   X'C024A71F'  IBM use only, Gets ARP
*                                      Entry
SIOCDARP             EQU   X'8024A720'  IBM use only, Deletes ARP
*                                      Entry
SIOCSHIWAT           EQU   X'8004A700'  Set High Water Mark
*                                         (Not Supported)
SIOCGHIWAT           EQU   X'4004A701'  Get High Water Mark
*                                         (Not Supported)
SIOCSLOWAT           EQU   X'8004A702'  Set Low Water Mark
*                                         (Not Supported)
SIOCGLOWAT           EQU   X'4004A703'  Get Low Water Mark
*                                         (Not Supported)
FIOFCTLNBIO          EQU   X'0000E402'  change blocking/nonblocking
*
IOCC#EDITACL         EQU   X'2000C100'  Edit ACL
IOCC#ILINK           EQU   X'4004E21A'  I_LINK
*  Constants for argument when FIONBIO is specified
IOCC#BLOCK           EQU   X'00000000'  Allow blocking to occur
IOCC#NONBLOCK        EQU   X'00000001'  Do not allow blocking to occur
*********************************************************************
* Packet mode or Extended Packet mode data record control data.
*                                                                   *
* Returned on master read when no control information is pending.   *
* In packet mode one byte is returned.  In extended packet mode, four *
* bytes are returned.  Data follows the control data.               *
*********************************************************************
TIOC_DATA            EQU   X'00'        Data packet
*********************************************************************
* Packet mode control byte - returned on master read()
```

```
*                                                                     *
* A single control byte is returned in packet mode.  In extended      *
* packet mode, four bytes are returned, with the non-extended bits    *
* in the fourth byte.  The equates below can be used against the      *
* fourth byte (with TM, OI and NI) or against all four bytes (with    *
* OC, NC, etc.).                                                      *
***********************************************************************
TIOCPKT_FLUSHREAD    EQU   X'01'        Input was flushed
TIOCPKT_FLUSHWRITE   EQU   X'02'        Output was flushed
TIOCPKT_STOP         EQU   X'04'        Stop output
TIOCPKT_START        EQU   X'08'        Start output
TIOCPKT_NOSTOP       EQU   X'10'        STOP/START not standard
TIOCPKT_DOSTOP       EQU   X'20'        STOP/START standard
***********************************************************************
* Extended Packet mode control byte - returned on master read()
***********************************************************************
TIOCXPKT_PASSTHRU    EQU   X'00000100'  3270 Passthrough mode
TIOCXPKT_NOPASSTHRU  EQU   X'00000200'  Not 3270 Passthrough mode
TIOCXPKT_ECHO        EQU   X'00000400'  ECHO set on
TIOCXPKT_NOECHO      EQU   X'00000800'  ECHO set off
TIOCXPKT_CHCP        EQU   X'00001000'  Code page change
TIOCXPKT_PWBEGIN     EQU   X'00002000'  Begin secure data
TIOCXPKT_PWEND       EQU   X'00004000'  End   secure data
***********************************************************************
* UPDTOFTE
***********************************************************************
IOCC#UPDTOFTE        EQU   20         UPDATE OFTE CMD
IOCUOFTE             DSECT ,          ARGUMENT BUFFER
IOCUOCMD             DS     F         SUBCMD
IOCUO#READ           EQU    1           READ
IOCUO#WRITE          EQU    2           WRITE
IOCUO#CS             EQU    3           COMPARE & SWAP
IOCUOVALUEBUFF       DS     0F        VALUE TO/FROM STATE AREA
IOCUOVOFFSET         DS     F            OFFSET  (>=0)
IOCUOVLEN            DS     F            LENGTH  (>0)
IOCUOVDATA           DS     0C           DATA
IOCUOCSBUFF          DSECT ,          COMPARE VALUE FOR CS SUBCMD
IOCUOCSOFFSET        DS     CL4          OFFSET (BYTE BDY)
IOCUOCSLEN           DS     CL4          LENGTH (BYTE BDY)
IOCUOCSDATA          DS     0C           DATA
*
IOCC#REGFILEINT      EQU   21         REGISTER FILE INTR
IOCC#FASTPATH        EQU   22         Set FastPath Ops
** BPXYIOCC End
```

BPXYIOV — Map the I/O Vector Structure



Purpose

Use the BPXYIOV macro to map the socket I/O vector structure used by the readv (BPX1RDV), writev (BPX1WRV), sendmsg (BPX2SMS), and recvmsg (BPX2RMS) callable services.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

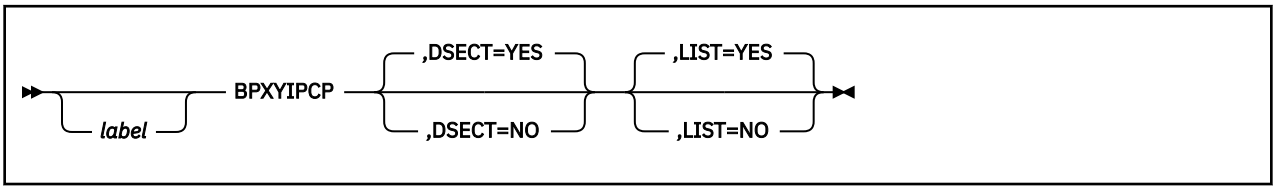
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYIOV macro expands as follows:

```
            BPXYIOV    ,
** BPXYIOV: Socket I/O Vectors
**  Used By: FCT OPN
IOV                 DSECT ,
IOV_ENTRY           DS    0F
IOV_BASE            DS    A      Address of buffer
IOV_LEN             DS    F      Length of buffer
*                                MSG_FLAGS must never be < 0
IOV#LENGTH          EQU   *-IOV_ENTRY Length of this structure
** BPXYIOV End
```

BPXYIPCP — Map Interprocess Communications Permissions



Purpose

Use the BPXYIPCP macro to map the data structure for interprocess communications permissions and other constants used by OpenExtensions callable services.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

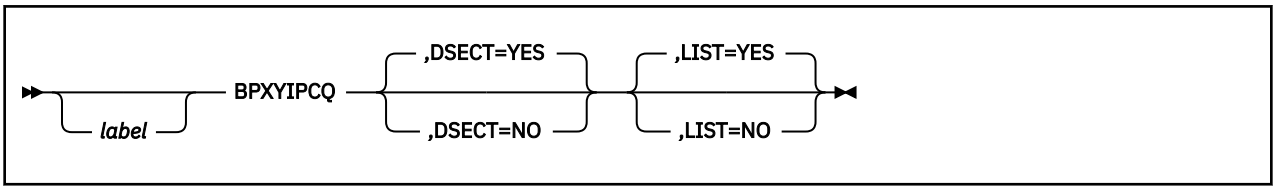
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYIPCP macro expands as follows:

```
          BPXYIPCP   ,
** BPXYIPCP: Interprocess Communications Permission
**  Used By: MCT, MGT, SCT, SGT, QCT, QGT
IPC_PERM             DSECT ,      Interprocess Communications
IPC_UID              DS    F      Owner's effective user ID
IPC_GID              DS    F      Owner's effective group ID
IPC_CUID             DS    F      Creator's effective user ID
IPC_CGID             DS    F      Creator's effective group ID
IPC_MODE             DS    XL4    Mode, mapped by BPXYMODE
IPC#LENGTH   EQU   *-IPC_PERM  Length of Interprocess Control block
* Key:
IPC_PRIVATE          EQU   0      Private key.
* Mode bits:                      Map over S_TYPE in BPXYMODE
IPC_CREAT            EQU   1      Create entry if key does not exist.
IPC_EXCL             EQU   2      Fail if key exists.
* Flag bits - semop, msgrcv, msgsnd:
IPC_NOWAIT           EQU   1      Error if request must wait.
* Control Command:
IPC_RMID             EQU   1      Remove identifier.
IPC_SET              EQU   2      Set options.
IPC_STAT             EQU   3      Access status.
* CONSTANTS WHICH MAP OVER BYTE S_TYPE, SEE BPXYMODE
** BPXYIPCP End
```

BPXYIPCQ — Map the Data Structure and Constants for the w_getipc Service



Purpose

Use the BPXYIPCQ macro to map the data structure and constants used by the w_getipc (BPX1GET) callable service.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

Usage Notes

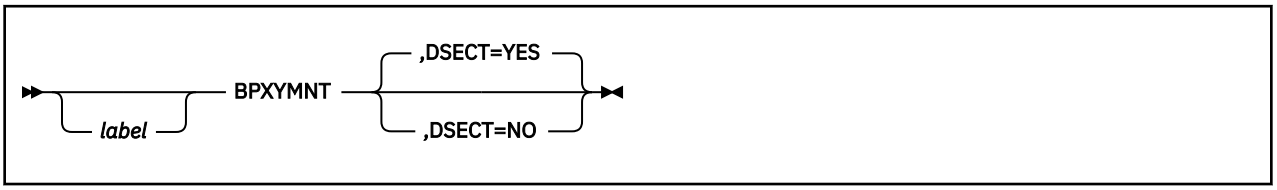
1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYIPCQ macro expands as follows:

```
          BPXYIPCQ   ,
** BPXYIPCQ: w_getipc interface mapping
**  Used By: GET
IPCQ          DSECT ,      Interprocess Communications - Query
IPCQLENGTH    DS    F      IPCQ#LENGTH used by system call.  If not
*                          equal, check BPXYIPCQ and system levels.
IPCQTYPE      DS    CL4    "IMSG", "ISEM", "ISHM", "OVER"
IPCQOVER      DS    0F     OVERVIEW MAPPING STARTS HERE
IPCQMID       DS    FL4    MEMBER ID
IPCQKEY       DS    XL4    KEY
IPCQIPCP      DS    CL20   MAPPED BY BPXYIPCP
IPCQGTIME     DS    XL4    TIME_T OF LAST ...GET()
IPCQCTIME     DS    XL4    TIME_T OF LAST ...CTL()
IPCQTTIME     DS    XL4    TIME_T CHANGED BY TERMINATION
IPCQREST      DS    0C     IPCQMSG, IPCQSHM, IPCQSEM
              ORG   IPCQREST Message Queue unique data
              DS    0F
IPCQBYTES     DS    F      # BYTES OF MESSAGES ON QUEUE
IPCQQBYTES    DS    F      MAX # BYTES OF MESSAGES ALLOWED ON QUEUE
IPCQLSPID     DS    F      PID OF LAST MSGSND()
IPCQLRPID     DS    F      PID OF LAST MSGRCV()
IPCQSTIME     DS    F      TIME_T OF LAST MSGSND()
IPCQRTIME     DS    F      TIME_T OF LAST MSGRCV()
IPCQNUM       DS    F      # OF MESSAGES ON QUEUE
```

```
IPCQRCNT      DS    F       COUNT OF WAITING MSGRCV
IPCQSCNT      DS    F       COUNT OF WAITING MSGSND
              DS    0CL16   MSGRCV AND MSGSND WAITERS
              DS    0CL8     MSGRCV - WAIT FOR TYPE
IPCQQRPID     DS    F         PROCESS ID
IPCQQRMSGTYPE DS    F         MESSAGE TYPE
              DS    0CL8    MSGSND - WAIT FOR ROOM TO SEND
IPCQQSPID     DS    F         PROCESS ID
IPCQQSMSGLEN  DS    F         MESSAGE LENGTH
              DS    9CL16   MSGSND AND MSGRCV WAITERS
              ORG   IPCQREST Semaphore Unique data
              DS    0F
IPCQLOPID     DS    XL4     PID OF LAST SEMOP
IPCQOTIME     DS    F       TIME_T LAST SEMOP
IPCQADJBADCNT DS    F       TERMINATION BUMPS SEM_VAL LIMITS
IPCQNSEMS     DS    FL2     NUMBER OF SEMAPHORES IN THIS SET
IPCQADJCNT    DS    FL2     NUMBER OF UNDO STRUCTURES
IPCQNCNT      DS    FL2     COUNT OF WAITERS FOR >0
IPCQZCNT      DS    FL2     COUNT OF WAITERS FOR =0
              DS    0CL16   WAITERS AND ADJUSTERS
              DS    0CL8     WAITER
IPCQSWPID     DS    F         PROCESS ID
IPCQSWNUM     DS    H         SEMAPHORE NUMBER
IPCQSWOP      DS    H         SEMAPHORE OPERATION
              DS    0CL8    ADJUSTER
IPCQSAPID     DS    F         PROCESS ID
IPCQSANUM     DS    H         SEMAPHORE NUMBER
IPCQSAADJ     DS    H         SEMAPHORE OPERATION
              DS    9CL16   WAITERS AND ADJUSTERS
              ORG   IPCQREST Shared Memory unique data
              DS    0F
IPCQACNT      DS    F       USE COUNT (#SHMAT - #SHMDT)
IPCQSEGSZ     DS    F       MEMORY SEGMENT SIZE
IPCQDTIME     DS    F       TIME_T OF LAST SHMDT()
IPCQATIME     DS    F       TIME_T OF LAST SHMAT()
IPCQLPID      DS    F       PID OF LAST SHMAT() OR SHMDT()
IPCQCPID      DS    XL4     PID OF CREATOR
IPCQATPID     DS    F       ATTACHED PROCESS ID
IPCQATADDRESS DS    F       SEGMENT ADDRESS FOR PROCESS
              DS    18F     MORE ATTACHED PROCESS IDS AND
*                           SEGMENT ADDRESS
              ORG   IPCQOVER Overview
                 DS    0F  MESSAGE QUEUES
IPCQOMSGNIDS     DS    F         Maximum number MSQs allowed
IPCQOMSGHIGHH2O  DS    F         Most MSQs at one time
IPCQOMSGFREE     DS    F         Number MSQs available
IPCQOMSGPRIVATE  DS    F         Number MSQs with Ipc_PRIVATE
IPCQOMSGKEYED    DS    F         Number MSQs with KEYs
IPCQOMSGREJECTS  DS    F         TIMES MSGGET DENIED
IPCQOMSGQBYTES   DS    F         MAX BYTES PER QUEUE
IPCQOMSGQMNUM    DS    F         MAX NUMBER MESSAGES PER QUEUE
IPCQOMSGNOALC    DS    F         # MSGSNDS THAT RETURNED ENOMEM
                 DS    F
                 DS    0F  SEMAPHORE
IPCQOSEMNIDS     DS    F         Maximum number SEMs allowed
IPCQOSEMHIGHH2O  DS    F         Most SEMs at one time
IPCQOSEMFREE     DS    F         Number SEMs available
IPCQOSEMPRIVATE  DS    F         Number SEMs with Ipc_PRIVATE
IPCQOSEMKEYED    DS    F         Number SEMs with KEYs
IPCQOSEMREJECTS  DS    F         TIMES SEMGET DENIED
IPCQOSEMSNSEMS   DS    F         MAX NUMBER OF SEMAPHORES PER SET
IPCQOSEMSNOPS    DS    F         MAX NUMBER OPERATION IN SEMOP
IPCQOSEMSBYTES   DS    F         STORAGE LIMIT
IPCQOSEMCBYTES   DS    F         STORAGE COUNT
                 DS    F
                 DS    0F  SHARED MEMORY
IPCQOSHMNIDS     DS    F         Maximum number SHMs allowed
IPCQOSHMHIGHH2O  DS    F         Most SHMs at one time
IPCQOSHMFREE     DS    F         Number SHMs available
IPCQOSHMPRIVATE  DS    F         Number SHMs with Ipc_PRIVATE
IPCQOSHMKEYED    DS    F         Number SHMs with KEYs
IPCQOSHMREJECTS  DS    F         TIMES SHMGET DENIED
IPCQOSHMSPAGES   DS    F         MAX # PAGES PER SYSTEM LIMIT
IPCQOSHMMPAGES   DS    F         MAX # PAGES PER SEGMENT LIMIT
IPCQOSHMNSEGS    DS    F         MAX # SEGMENTS PER PROCESS LIMIT
IPCQOSHMCPAGES   DS    F         CURRENT # BYTES SYSTEM WIDE
IPCQOSHMBIGGEST  DS    F         LARGEST SEGMENT ALLOCATED
              ORG   ,
IPCQ#LENGTH   EQU   *-IPCQ Storage needed for w_getipc function
* w-getipc Command:
IPCQ#MSG      EQU   1       Retrieve next message queue
IPCQ#SHM      EQU   2       Retrieve next shared memory segment
```

```
IPCQ#SEM      EQU   3      Retrieve next semaphore set
IPCQ#ALL      EQU   4      Retrieve next member, all mechanisms
IPCQ#OVER     EQU   5      Retrieve overview
** BPXYIPCQ End
```

BPXYMNT — Map the File System Parameters for the mount Service



Purpose

Use the BPXYMNT macro to map file-system-specific parameters for the mount (BPX1MNT) callable services.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

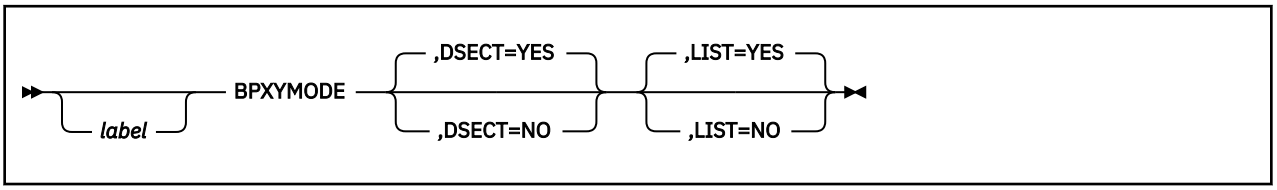
Usage Notes

1. The BPXYMNT mapping macro expands as follows:

```
         BPXYMNT  ,
MNT               DSECT  ,
MNT_MAP_VERSION   DC   AL2(MNTMAPVER) Version number
MNT_NETRC_FLAG    DS   XL1            NETRC file usage flag
MNT_NETRC         EQU  0
MNT_NONETRC       EQU  1
*
MNT_TRANS_FLAG    DS   XL1            Translation Flag Byte
MNT_TRANS_LIST    EQU  0              File Extension List translation
MNT_TRANS_ALL     EQU  1              Translate all file data
MNT_TRANS_NO      EQU  2              Do not translate file data
*
MNT_TRANS_TABLE   DS   CL8            Translation table name
*                                     X'0000000000000000' for default
*
MNT_ATTRCACHE     DS   XL1            Attribute caching
MNT_ATTRCACHE_YES EQU  0
MNT_ATTRCACHE_NO  EQU  1
*
MNT_VERSION       DS   F              Protocol Version to use.
MNT_VERSION_NOT_SPEC EQU 0
MNT_VERSION_2     EQU  2
MNT_VERSION_3     EQU  3
*
MNT_PROTOCOL      DS   XL1            Communication Protocol
MNT_PROTOCOL_TCP  EQU  0              TCP
MNT_PROTOCOL_UDP  EQU  1              UDP
*
MNT_PAD1          DS   XL1            Padding
*
MNT_USER_LEN      DS   F              Mount User ID Length
MNT_USER_PTR      DS   A(MNT_USERID)  Mount User ID
*
MNT_PASSWD_LEN    DS   F              Mount Password Length
MNT_PASSWD_PTR    DS   A(MNT_PASSWD)  Mount Password
```

```
*
MNT_ATTRMAX        DS   F              Maximum lifetime of cached
*                                     attributes in seconds.
MNT_ATTRMAX_DEFAULT EQU 60
MNT_READ_AHEAD     DS   F              Maximum number of disk blocks
*                                     to read ahead
MNT_READ_AHEAD_DEFAULT EQU 1
*
MNT_RETRY          DS   F              Number of times to resend to
*                                     NFS server. Specify -1 to
*                                     retry forever.
MNT_RETRY_DEFAULT  EQU  3
*
MNT_TIMEOUT        DS   F              Time to wait for response from
*                                     NFS server, in tenths of second.
MNT_TIMEOUT_DEFAULT EQU 7
*
MNT_PORT_ANY       EQU  0              Port any
                   DS   XL6            Pad to doubleword
MNTMAPVER          EQU  MNTMAPVER02    Current version
MNTMAPVER01        EQU  1
MNTMAPVER02        EQU  2              Current version
```

BPXYMODE — Map Mode Constants



Purpose

Use the BPXYMODE macro to map the mode constants used by OpenExtensions callable services.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

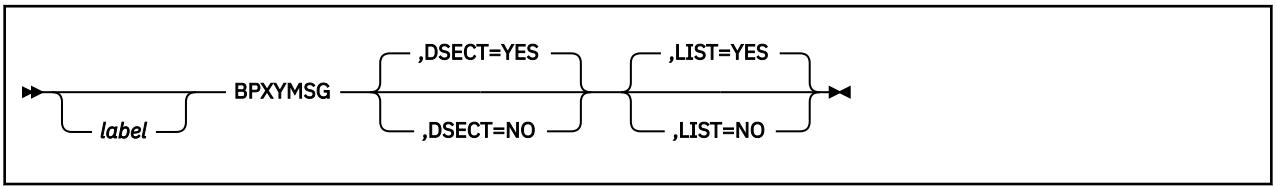
Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYMODE mapping macro expands as follows:

```
         BPXYMODE   ,
S_MODE                DSECT ,
                      DS    0F
*
S_TYPE                DS    B       File types, mapped by BPXYFTYP
*                                   Flag bytes
S_SUBTYPE             DS    B       Subtype for external links
*
S_MODE2               DS    B       Flag byte 2
S_RES02               DS    0BL4    Reserved for IBM use
*                                   Set ID flags
S_ISUID               EQU   X'08'   Set user ID on execution
S_ISGID               EQU   X'04'   Set group ID on execution
S_ISVTX               EQU   X'02'   Keep loaded executable in
                                    storage (sticky bit)
*                                   Owner flags
S_IRWXU1              EQU   X'01'   All permissions for user - part I
S_IRUSR               EQU   X'01'   Read permission
*
S_MODE3               DS    B       Flag byte 3
*                                   Owner flags - continued
S_IRWXU2              EQU   X'C0'   All permissions for user - Part II
S_IWUSR               EQU   X'80'   Write permission
S_IXUSR               EQU   X'40'   Search (if a directory) or
*                                   execute (otherwise) permission
*                                   Group flags
S_IRWXG               EQU   X'38'   All permissions for group
S_IRGRP               EQU   X'20'   Read permission
```

```
S_IWGRP              EQU   X'10'  Write permission
S_IXGRP              EQU   X'08'  Search (if a directory) or
*                                 execute (otherwise) permission
*                                 Other flags
S_IRWXO              EQU   X'07'  All permissions for other
S_IROTH              EQU   X'04'  Read permission
S_IWOTH              EQU   X'02'  Write permission
S_IXOTH              EQU   X'01'  Search (if a directory) or
*                                 execute (otherwise) permission
S_MODE#LENGTH        EQU   *-S_MODE Length this structure
```

BPXYMSG — Map Interprocess Communications Message Queues



Purpose

Use the BPXYMSG macro to map the data structures and constants for the OpenExtensionscallable services that create and control interprocess communications message queues.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

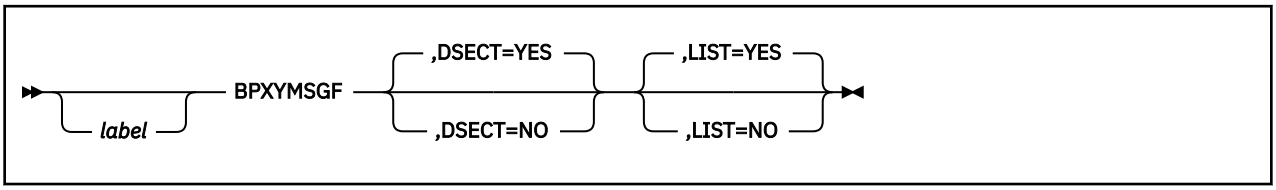
Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The MSGBUF DSECT is generated with either DSECT=YES or DSECT=NO. If you specify DSECT=NO, you may need an additional DSECT or CSECT statement to return to the current DSECT or CSECT.
3. The BPXYMSG macro expands as follows:

```
          BPXYMSG    ,
** BPXYMSG: Interprocess Communication Message Queue Structure
**  Used By: msgctl
MSQID_DS             DSECT ,      message queue structure
MSG_PERM             DS    CL(IPC#LENGTH) Mapped by BPXYIPCP
MSG_QNUM             DS    F      # of messages on queue
MSG_QBYTES           DS    F      max bytes allowed on queue
MSG_LSPID            DS    F      process ID of last msgsnd()
MSG_LRPID            DS    F      process ID of last msgrcv()
MSG_STIME            DS    F      time of last msgsnd()
MSG_RTIME            DS    F      time of last msgrcv()
MSG_CTIME            DS    F      time of last change  get/ctl
MSQ#LENGTH           EQU *-MSQID_DS Length of this DSECT
MSGBUF               DSECT ,      Message buffer - msgsnd, msgrcv
MSG_TYPE             DS    F      Message type
MSG_MTEXT            DS    CL100  Message text
MSGB#LENGTH          EQU *-MSGBUF Length of this DSECT
MSGXBUF              DSECT ,      Message buffer - msgxrcv
MSGX_MTIME           DS    F      time message sent
MSGX_UID             DS    F      sender's effective UID
MSGX_GID             DS    F      sender's effective GID
MSGX_PID             DS    F      sender's PID
MSGX_TYPE            DS    F      Message type
MSGX_MTEXT           DS    CL100  Message text
```

```
MSGX#LENGTH          EQU *-MSGXBUF Length of this DSECT
* Flag bits - msgrcv (also IPC_NOWAIT
MSG_NOERROR          EQU   4      No error if big message.
MSG_INFO             EQU   8      Use MSGXBUF not MSGBUF format
** BPXYMSG End
```

BPXYMSGF — Map the Message Flags



Purpose

Use the BPXYMSGF macro to map the message flags used by the send (BPX1SND), recv (BPX1RCV), sendto (BPX1STO), recvfrom (BPX1RFM), sendmsg (BPX2SMS), and recvmsg (BPX2RMS) callable services.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

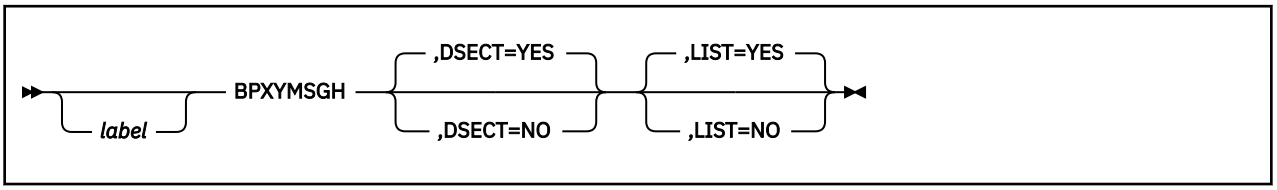
Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYMSGF macro expands as follows:

```
            BPXYMSGF   ,
** BPXYMSGF: Socket access flags
**  Used By: FCT OPN
MSG_FLAGS            DSECT ,
MSG_FLAGS1           DS    B      I_flags - byte 1
MSGFHIGH             EQU   X'80'  DO NOT USE THIS BIT!
*                                 MSG_FLAGS must never be < 0
MSG_ACK_GEN          EQU   X'40'  Generate a UDP 'ACK packet'
*                                 automatically to the originator
*                                 if an incoming UDP packet arrives
*
MSG_ACK_TIMEOUT      EQU   X'20'  The caller expects an incoming UDP
*                                 packet within the "standard ACK
*                                 time interval".  Return to caller
*                                 with an EINTR return code if no
*                                 incoming UDP packet arrives
*                                 within this time interval.
MSG_ACK_EXPECTED     EQU   X'10'  (Used along with MSG_ACK_TIMEOUT)
*                                 The incoming packet is expected to
*                                 be an ACK. If the ACK arrives,
*                                 the caller does not need to be
*                                 activated to process it.
*                                 Instead, the protocol will just
*                                 cancel the timeout and let the
*                                 application wait for the real data
```

```
*                                  to arrive.
MSG_FLAGS2           DS    B       MSG_flags - byte 2
*
MSG_FLAGS3           DS    B       MSG_flags - byte 3
MSG_EOF              EQU   X'80'   Close after send
MSG_FLAGS4           DS    B       MSG_flags - byte 4
MSG_WAITALL          EQU   X'40'   Wait until all data returned
MSG_CTRUNC           EQU   X'20'   Control data truncated
MSG_TRUNC            EQU   X'10'   Normal data truncated
MSG_EOR              EQU   X'08'   Terminate a record
MSG_DONTROUTE        EQU   X'04'   Send without network routing
MSG_PEEK             EQU   X'02'   Peek at incoming data
MSG_OOB              EQU   X'01'   Receive out of band data
MSG#LENGTH          EQU   *-MSG_FLAGS Length of this structure
** BPXYMSGF End
```

BPXYMSGH — Map the Message Headers



Purpose

Use the BPXYMSGH macro to map the message headers used by the sendmsg (BPX2SMS) and recvmsg (BPX2RMS) callable services.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

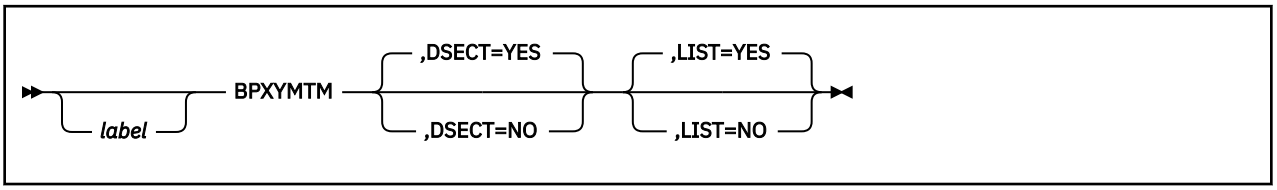
Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYMSGH macro expands as follows:

```
            BPXYMSGH   ,
** BPXYMSGH: MSGH system call structure
**  Used By: SendMsg / RecvMsg
MSGH                DSECT ,
MSGHBEGIN           DS    0D
*
MSGHNAMEPTR         DS    A(0)   Pointer to a structure that contains
*                                 the recipient's address.
MSGHNAMELEN         DS    F'0'   Buffer length.
MSGHIOVPTR          DS    A(0)   Pointer to an array of IOVEC buffers.
MSGHIOVNUM          DS    F'0'   Number of elements in IOVEC array.
MSGHCONTROLPTR      DS    0AL4   Pointer to ancillary data buffer
MSGHACCRIGHTSPTR    DS    A(0)   Pointer to access rights buffer.
MSGHCONTROLLEN      DS    0FL4   Length of ancillary data buffer
MSGHACCRIGHTSLEN    DS    F'0'   Access rights buffer length.
MSGHFLAGS           DS    F'0'   Output flags on received message
*
*   Constants
*
MSGH#LENGTH         EQU   *-MSGH   Length of MsgH
*
CMSGPTR             DS    A(0)   CMsg pointer
*
CMSGHDR             DSECT ,
CMSGLEN             DS    F'0'   Length, including header
CMSGLEVEL           DS    F'0'   Level
CMSGTYPE            DS    F'0'   Type
```

```
CMSGDATA             DS    0C     Data
*
*   Constants
*
SCM_RIGHTS           EQU   1        Access Rights
*
** BPXYMSGH End
```

BPXYMTM — Map the Modes for the mount and umount Services



Purpose

Use the BPXYMTM macro to map the modes for the mount (BPX1MNT) and umount (BPX1UMT) callable services.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

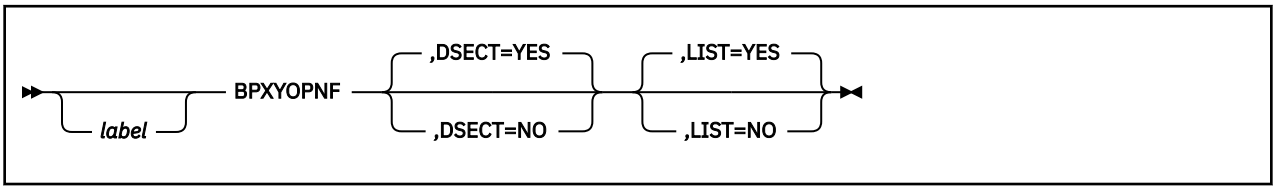
Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYMTM mapping macro expands as follows:

```
         BPXYMTM    ,
MTM                 DSECT ,
MTM1                DS    B     Flag byte 1
MTMRO               EQU   X'80' Mount file set read-only
MTMRDWR             EQU   X'40' Mount file set read/write
MTM1RES20           EQU   X'20' Must not be used
MTMUMOUNT           EQU   X'10' This is a normal unmount request.
*                               If no one is using any of the files
*                               in the named filesystem, the unmount
*                               is done.  Otherwise, the request is
*                               rejected.
MTM1RES08           EQU   X'08' Must not be used
MTM1RES04           EQU   X'04' Must not be used
MTM1RES02           EQU   X'02' Must not be used
MTM1RES01           EQU   X'01' Must not be used
MTM2                DS    B     Flag byte 2
MTM2RES80           EQU   X'80' Must not be used
MTM2RES40           EQU   X'40' Must not be used
MTM2RES20           EQU   X'20' Must not be used
MTM2RES10           EQU   X'10' Must not be used
MTM2RES08           EQU   X'08' Must not be used
MTM2RES04           EQU   X'04' Must not be used
MTM2RES02           EQU   X'02' Must not be used
MTM2RES01           EQU   X'01' Must not be used
MTM3                DS    B     Flag byte 3 - Reserved for IBM use
MTM3RES80           EQU   X'80' Must not be used
MTM3RES40           EQU   X'40' Must not be used
```

```
MTM4                 DS    B     Flag byte 4 - Reserved for IBM use
MTM#LENGTH           EQU   *-MTM Length of this structure
```

BPXYOPNF — Map Flag Values for the open and fcntl Services



Purpose

Use the BPXYOPNF macro to map flag values for the open (BPX1OPN) and fcntl (BPX1FCT) callable services.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

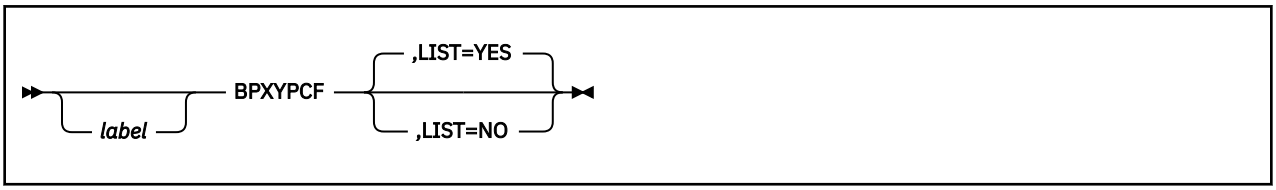
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYOPNF mapping macro expands as follows:

```
         BPXYOPNF   ,
O_FLAGS              DSECT ,
O_FLAGS1             DS    B      Open flags - byte 1
OPNFHIGH             EQU   X'80'  DO NOT USE THIS BIT!
*                                 O_FLAGS must never be < 0
O_FLAGS2             DS    B      Open flags - byte 2
OPNFEXEC             EQU   X'80'  Execute access requested -
*                                 authorization required for use
O_FLAGS3             DS    B      Open flags - byte 3
O_ASYNCSIG           EQU   X'02'  An asynchronous signal may occur
O_SYNC               EQU   X'01'  Force synchronous updates
O_FLAGS4             DS    B      Open flags - byte 4
O_CREXCL             EQU   X'C0'  Create file only if non-existent
O_CREAT              EQU   X'80'  Create file
O_EXCL               EQU   X'40'  Exclusive flag
O_NOCTTY             EQU   X'20'  Not a controlling terminal
O_TRUNC              EQU   X'10'  Truncate flag
O_APPEND             EQU   X'08'  Set offset to EOF on write
O_NONBLOCK           EQU   X'04'  Don't block this file
O_RDWR               EQU   X'03'  Open for Read and Write
O_RDONLY             EQU   X'02'  Open for Read Only
O_WRONLY             EQU   X'01'  Open for Write Only
O_ACCMODE            EQU   X'03'  Mask for file access modes
O_GETFL              EQU   X'0F'  Mask for file access modes and
*                                 file status flags together
OPNF#LENGTH          EQU   *-O_FLAGS Length of this structure
```

BPXYPCF — Map Command Values for the pathconf and fpathconf Services



Purpose

Use the BPXYPCF macro to map the command values for the pathconf (BPX1PCF) and fpathconf (BPX1FPC) callable services.

Parameters

label
is an optional assembler label for the statement.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

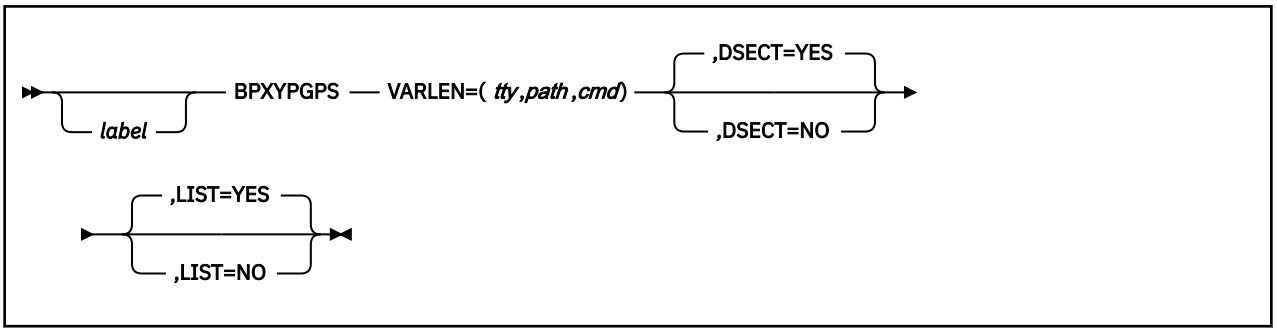
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The DSECT= parameter is allowed but ignored.
2. The PRINT OFF assembler statement overrides LIST=YES.
3. The BPXYPCF mapping macro expands as follows:

```
         BPXYPCF    ,
PC_CHOWN_RESTRICTED  EQU 1        _POSIX_CHOWN_RESTRICTED option
PC_LINK_MAX          EQU 2        LINK_MAX option
PC_MAX_CANON         EQU 3        _POSIX_MAX_CANON option
PC_MAX_INPUT         EQU 4        _POSIX_MAX_INPUT option
PC_NAME_MAX          EQU 5        NAME_MAX option
PC_NO_TRUNC          EQU 6        _POSIX_NO_TRUNC option
PC_PATH_MAX          EQU 7        PATH_MAX option
PC_PIPE_BUF          EQU 8        PIPE_BUF option
PC_VDISABLE          EQU 9        _POSIX_VDISABLE option
```

BPXYPGPS — Map the Response Structure for the w_getpsent Service



Purpose

Use the BPXYPGPS macro to map the response structure for the w_getpsent (BPX1GPS) callable service.

Parameters

label
is an optional assembler label for the statement.

VARLEN=(*tty*,*path*,*cmd*)
describes the number of bytes needed to map:

- The controlling TTY name and its length
- The path name and its length
- The command and its length

If a parameter is omitted, it defaults to the maximum (1028 bytes). Specify `0` if the associated field is not needed.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

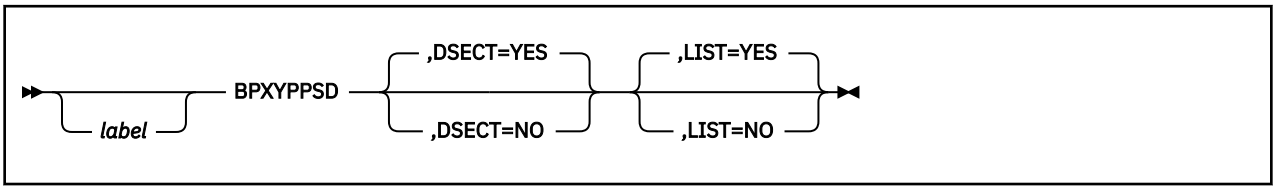
Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYPGPS macro expands as follows:

```
         BPXYPGPS   VARLEN=(1028,1028,1028)
PGPS                 DSECT ,
PGPSSTATUS0          DS    B         CMS status
PGPSSWAP             EQU   X'80'       Swapped out
*                    EQU   X'7F'       Not used
PGPSSTATUS1          DS    B         Process status
PGPSSTOPPED          EQU   X'80'       Stopped process
```

```
PGPSTRACE             EQU   X'40'       Reserved
PGPSMULTHREAD         EQU   X'20'        0=One OpenExtensions active
PGPSPTHREAD           EQU   X'10'        0=No pthread task in process
*                     EQU   X'0F'       Not used
PGPSSTATUS2           DS    B          System Call Status
PGPSLENERR            EQU   X'80'       PGPSLENGTH conflict
*                     EQU   X'7F'       Not used
PGPSSTATUS3           DS    CL1       State of reported task - with
*                           PGPSPTHREAD=0 the most recent created thread
*                           PGPSPTHREAD=1 the initial pthread task (IPT)
PGPSZOMBIE            EQU   C'Z'        Process canceled
PGPSCHILD             EQU   C'W'        Waiting for child
PGPSFORK              EQU   C'X'        spawn a new process
PGPSSLEEP             EQU   C'S'        sleep issued
PGPSWAITC             EQU   C'C'        Communication kernel wait
PGPSWAITF             EQU   C'F'        File System kernel wait
PGPSWAITO             EQU   C'K'        Other kernel wait
PGPSRUN               EQU   C'R'        Not in kernel wait, running
PGPSPID               DS    F          Process ID
PGPSPPID              DS    F          Parent ID
PGPSSID               DS    F          Session ID (leader)
PGPSPGPID             DS    F          Process Group
PGPSFGPID             DS    F          Foreground Process Group
PGPSEUID              DS    F          Effective User ID
PGPSRUID              DS    F          Real User ID
PGPSSUID              DS    F          Saved Set User ID
PGPSEGID              DS    F          Effective Group ID
PGPSRGID              DS    F          Real Group ID
PGPSSGID              DS    F          Saved Set Group ID
PGPSTSIZE             DS    F          Total size
PGPSSTARTTIME         DS    F          Starting time, GMT since EPOCH
PGPSUSERTIME          DS    F          User CPU time (clock_t)
PGPSSYSTIME           DS    F          System CPU time (clock_t)
PGPSCONTTYBLEN        DC    A(1028)          L'PGPSCONTTYBUF
PGPSCONTTYPTR         DC    A(PGPSCONTTYBUF) ->PGPSCONTTYBUF
PGPSPATHBLEN          DC    A(1028)          L'PGPSPATHBUF
PGPSPATHPTR           DC    A(PGPSPATHBUF)   ->PGPSPATHBUF
PGPSCMDBLEN           DC    A(1028)          L'PGPSCMDBUF
PGPSCMDPTR            DC    A(PGPSCMDBUF)    ->PGPSCMDBUF
PGPS#LENGTH           EQU   *-PGPS    Length of this structure
* Variable portion - Controlling terminal buffer
PGPSCONTTYBUF         DS    0CL1028   ConTty Len+Buf
PGPSCONTTYLEN         DS    FL4       Length ConTty returned
PGPSCONTTY            DS    CL1024    ConTty
PGPSPATHBUF           DS    0CL1028   Pathname Len+Buf
PGPSPATHLEN           DS    FL4       Length Pathname returned
PGPSPATH              DS    CL1024    Pathname
PGPSCMDBUF            DS    0CL1028   Command Len+Buf
PGPSCMDLEN            DS    FL4       Length Command returned
PGPSCMD               DS    CL1024    Command
PGPS#STORAGE          EQU   *-PGPS    Length, total area used
```

BPXYPPSD — Map the Signal Delivery Data Structure



Purpose

Use the BPXYPPSD macro to map the signal delivery data structure passed to a signal interface routine (SIR) by OpenExtensions callable services.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

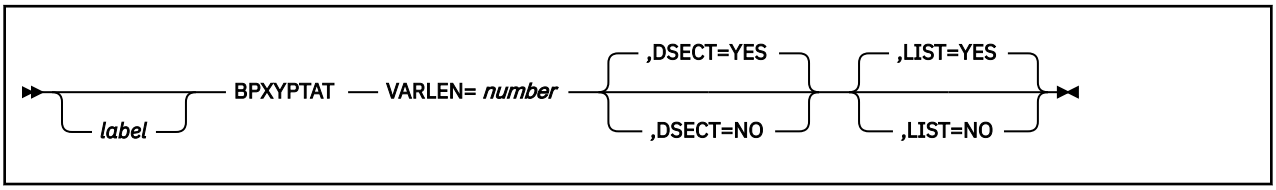
Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYPPSD mapping macro expands as follows:

```
         BPXYPPSD   ,
PPSD                DSECT ,
PPSDID              DC    C'PPSD'  Eye-catcher
PPSD#ID             EQU   C'PPSD'    Control Block Acronym
PPSDSP              DS    FL1      Subpool number of this PPSD
PPSD#SP             EQU   230        Subpool for the PPSD
PPSDLEN             DC    AL3(PPSD#LENGTH)   Length this structure
*
*    ****************************************************************
*     PpsdSIRParms is used to set up a parameter list to the
*     signal interface routine (SIR).  When the SIR is invoked, the
*     address of PpsdSIRParms field is set in Register 1.  The
*     PpsdAddrPpsd contains the address of the Ppsd.
*    ****************************************************************
*
PPSDSIRPARMS        DS    0A       SIR Parameters
PPSDADDRPPSD        DC    A(PPSD)  Pointer to the top of the Ppsd
PPSDSIRPARMEND      EQU   X'80'    End of Parameters flag set on
                    DS    F        Reserved for IBM use
PPSDSIGNUM          DS    F        Signal number
                    DS    FL2      Reserved for IBM use
PPSDACTION          DS    B        Action for this signal
*                   EQU   4           catch
*                   EQU   5           SIR determines
PPSDFLAGS           DS    B        X'0F' Reserved for IBM use
PPSDASYNC           EQU   X'80'    Signal is asynchronous
PPSDDUMP            EQU   X'40'    Dump for terminating signals
```

```
PPSDPTHREADKILL      EQU   X'20'    Signal sent via BPX1PTK
PPSDTHISTHREADGEN    EQU   X'10'    Sending=Receiving thread
PPSDSAHANDLER        DS    A        Addr of catcher function
PPSDSAMASK           DS    XL8      Signal mask to be used during
*                                   handler execution
PPSDSAFLAGS          DS    XL4      X'3FFFFFFF' Reserved for IBM use
PPSDNOCLDSTOP        EQU   X'80'     Do not generate SIGCHLD on stops
PPSDOLDSTYLE         EQU   X'40'     Signal defined by signal function
PPSDCURRENTMASK      DS    XL8      Current signal mask
PPSDSIR              DS    A        Addr Signal interrupt routine
PPSDUSERDATA         DS    A        User data field
PPSDGENREGS          DS    CL64     Users general regs at interrupt
PPSDPSW              DS    XL8      Users PSW
PPSDARREGS           DS    16F      Users AR regs
PPSDKILDATA          DS    FL2      User specified data on BPX1KIL
PPSDKILOPTS          DS    XL2      X'7FFF' Reserved for IBM use
*                                   User specified options on BPX1KIL
PPSDPTBYPASS         EQU   X'80'    Reserved
PPSDEND              DS    0D       End of PPSD on double word
PPSD#LENGTH          EQU   *-PPSD   Length of this structure
```

BPXYPTAT — Map Attributes for the pthread_create Service



Purpose

Use the BPXYPTAT macro to map the attributes for the pthread_create (BPX1PTC) callable service.

Parameters

label
is an optional assembler label for the statement.

VARLEN=*number*
defines the number of bytes set aside to define the pthread attributes.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

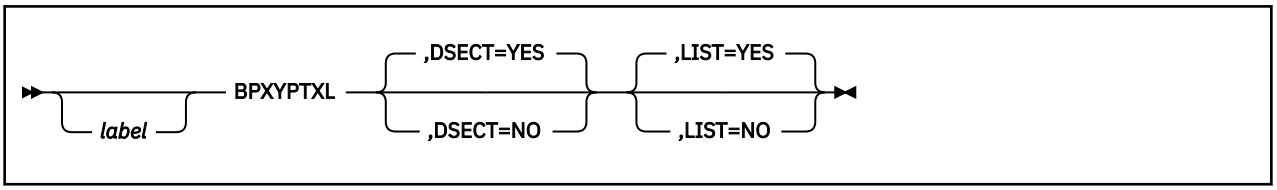
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYPTAT mapping macro expands as follows:

```
         BPXYPTAT   VARLEN=1024
PTAT                DSECT ,
PTATEYE             DC    C'BPXYPTAT'       Eye Catcher
PTATLENGTH          DC    A(PTAT#LENGTH)    Length of PTAT
PTATSYSOFFSET       DC    A(PTATSYSOFFVAL)  Offset of SYSATTRS
PTATSYSLENGTH       DC    A(PTATSYSLENVAL)  Length of SYSATTRS
PTATUSEROFFSET      DC    A(PTATUSEROFFVAL) Offset of USERATTRS
PTATUSERLENGTH      DC    A(L'PTATUSERATTRS) Length of USERATTRS
PTATSYSOFFVAL       EQU   *-PTAT Offset value of System Attribute Area
PTATSYSATTRS        DS    0F System attributes
PTATDETACHSTATE     DS    F  Detach State of thread to be created:
PTATUNDETACHED      EQU   0
PTATDETACHED        EQU   1
PTATWEIGHT          DS    F  Weight of thread to be created:
PTATHEAVY           EQU   0
PTATMEDIUM          EQU   1
PTATSYNCTYPE        DS    F  Synchronous processing type of thread:
PTATSYNCHRONOUS     EQU   0
                    DS    CL32   Reserved for IBM use
PTATSYSLENVAL       EQU   *-PTATSYSATTRS Length of System Attributes
PTATUSEROFFVAL      EQU   *-PTAT   Offset of user attribute area
PTATUSERATTRS       DS    CL1024   User attributes area
PTAT#LENGTH         EQU   *-PTAT   Length of this structure
```

BPXYPTXL — Map the Parameter List for the pthread_exit_and_get Service



Purpose

Use the BPXYPTXL macro to map the parameter list returned by the pthread_exit_and_get (BPX1PTX) callable service.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

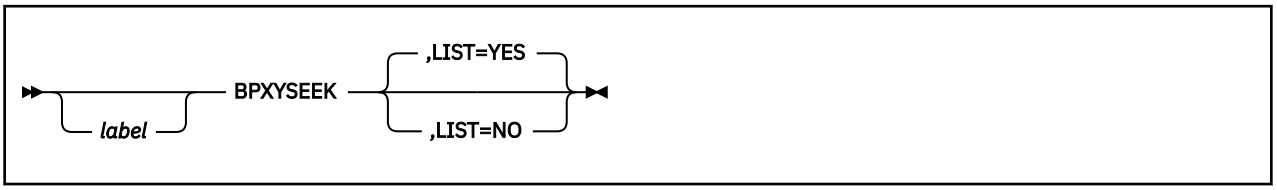
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYPTXL mapping macro expands as follows:

```
         BPXYPTXL   ,
PTXL                DSECT ,     Parameter List returned by BPX1PTX
PTXLWORKAREAPTR     DS    A     Pointer to User Work Area
PTXLATTRIBUTEPTR    DS    A     Pointer to User Attributes
PTXLTHIDPTR         DS    A     Pointer to Thread ID
PTXLSTATUSPTR       DS    A     Pointer to Thread Run Status
PTXL#LENGTH         EQU   *-PTXL
PTXLRS              DSECT ,     Thread Run Status
                    DS    0F
PTXLRSFLAGS         DS    0BL4  Thread Run Status Flags
PTXLRSFLAGS0        DS    B     1st byte
PTXLRSREADY         EQU   X'80'   Thread is ready to run
PTXLRSFLAGS1        DS    B     2nd byte
PTXLRSFLAGS2        DS    B     3rd byte
PTXLRSFLAGS3        DS    B     4th byte
PTXLRS#LENGTH       EQU   *-PTXLRS
```

BPXYSEEK — Map Constants for the lseek Service



Purpose

Use the BPXYSEEK macro to map the constants used by the lseek (BPX1LSK) callable service.

Parameters

label
is an optional assembler label for the statement.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

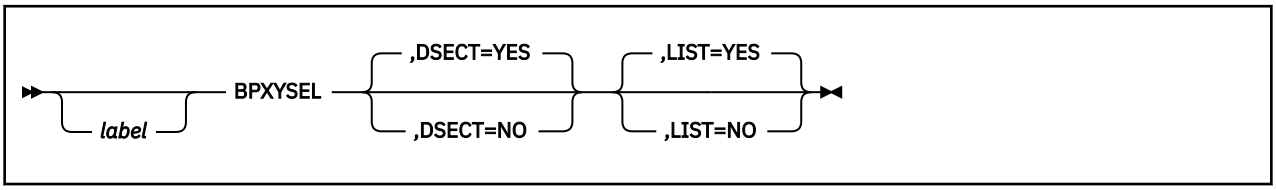
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The DSECT= parameter is allowed but ignored.
2. The PRINT OFF assembler statement overrides LIST=YES.
3. The BPXYSEEK mapping macro expands as follows:

```
         BPXYSEEK   ,
SEEK_SET             EQU   0    Set file offset to offset
SEEK_CUR             EQU   1    Set file offset to current + offset
SEEK_END             EQU   2    Set file offset to EOF + offset
```

BPXYSEL — Map Options for the select/selectex Service



Purpose

Use the BPXYSEL macro to map the options used by the select/selectex (BPX1SEL) callable service.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

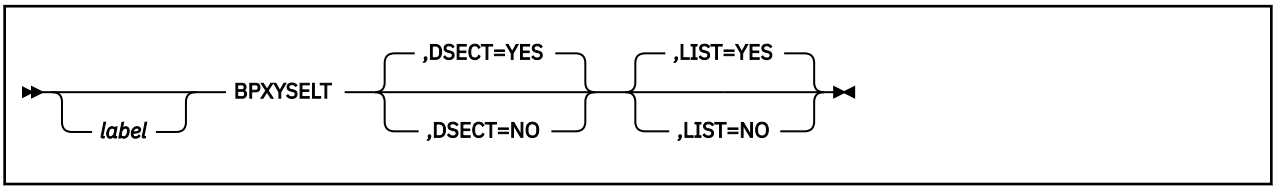
Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYSEL macro expands as follows:

```
            BPXYSEL    ,
** BPXYSEL: Select Options
**  Used By: SEL
SEL                 DSECT ,
SELBEGIN            DS    0F
*
SELBITS             DS    0XL4    Flag Bits.8F FF FF FF Reserved
SELPOLLFLAGS        DS    XL2     Select flags / Poll (r)events
*-----------------------------------------------------
* Select flags
*-----------------------------------------------------
SELFLAGS            ORG   SELPOLLFLAGS
                    DS    XL1
*                   EQU   X'80'   Never use this bit
SELREAD             EQU   X'40'   Descriptor ready for read.
SELWRITE            EQU   X'20'   Descriptor ready for write.
SELXCEPT            EQU   X'10'   Descriptor ready for exception.
                    DS    XL1     Available byte
*-----------------------------------------------------
* Poll Events/Returned Events
*-----------------------------------------------------
SELPOLLEVENTS       ORG   SELPOLLFLAGS
                    DS    XL2     Mapped by PollEvents(BPXYPOLL)
SELPOLLREVENTS      ORG   SELPOLLFLAGS
                    DS    XL2     Mapped by PollRevents(BPXYPOLL)
*
                    DS    XL1     Available byte
                    DS    XL1     Reserved for internal use
```

```
*
*   Constants
*
SEL#LENGTH           EQU   *-SEL   Length of SEL
SEL#QUERY            EQU   1       Query function
SEL#CANCEL           EQU   2       Cancel function
SEL#BATSELQ          EQU   3       Batch-Select Query function
SEL#BATSELC          EQU   4       Batch-Select Cancel function
SEL#POLLQUERY        EQU   5       Poll Query function
SEL#BATPOLLQ         EQU   6       Batch-Poll Query function
SEL#BATPOLLC         EQU   7       Batch-Poll Cancel function
SEL#POLLCANCEL       EQU   8       Poll Cancel function
SEL#BITSBACKWARD     EQU   0       Bit Backward Order by word
SEL#BITSFORWARD      EQU   1       Bit Forward Order by word
SEL#TYPES            EQU   3       3 TYPES (Read Write Except)
SEL#RBIT             EQU   64      Read bit position in byte
SEL#WBIT             EQU   32      Write bit position in byte
SEL#XBIT             EQU   16      Xcept bit position in byte
** BPXYSEL End
```

BPXYSELT — Map the Timeout Value for the select/selectex Service



Purpose

Use the BPXYSELT macro to map the timeout value for the select/selectex (BPX1SEL) callable service.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

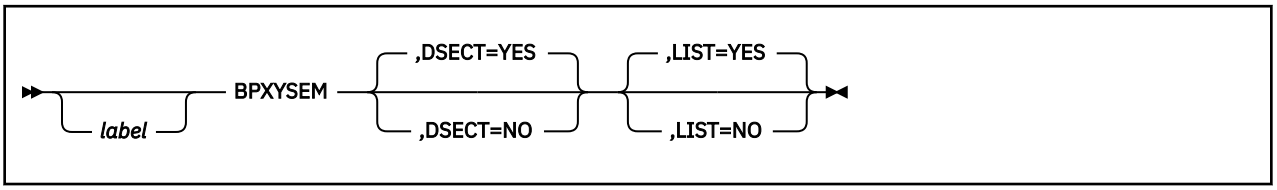
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYSELT macro expands as follows:

```
            BPXYSELT   ,
** BPXYSELT: Select Time Structure
**  Used By: Select Syscall
SELT                DSECT ,
SELTBEGIN           DS    0D
*
TV_SEC              DS    F'0'   Seconds
TV_USEC             DS    F'0'   Microseconds
*
*   Constants
*
SELT#LENGTH         EQU   *-SELT Length of SELT
** BPXYSELT End
```

BPXYSEM — Map Interprocess Communications Semaphores



Purpose

Use the BPXYSEM macro to map the data structures and constants for the OpenExtensionscallable services that create and control interprocess communications semaphores.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

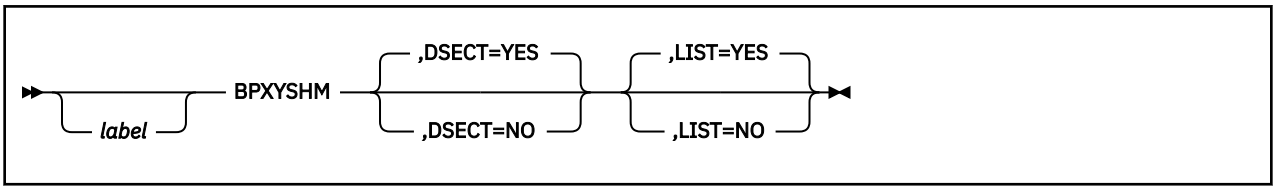
Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The SEMID_DS, SEM_ARRAY, and SEM_BUF_ELE DSECTs are generated with either DSECT=YES or DSECT=NO. If you specify DSECT=NO, you may need an additional DSECT or CSECT statement to return to the current DSECT or CSECT.
3. The BPXYSEM macro expands as follows:

```
          BPXYSEM    ,
** BPXYSEM: Interprocess Communications Permission
**  Used By: XSO, XSC
SEMID_DS             DSECT ,      semctl structure
SEM_PERM             DS    CL(IPC#LENGTH)   Mapped by BPXYIPCP
SEM_NSEMS            DS    H      number of semaphores in set
                     DS    H      spacer
SEM_OTIME            DS    FL4    last semop() time
SEM_CTIME            DS    FL4    last time changed by semctl()
SEM#LENGTH           EQU   *-SEMID_DS   Length of this DSECT
* SETVAL - a one element array for Semaphore_Number
* SETALL, GETALL - an array with Number_of_Semaphore elements
SEM_ARRAY            DSECT ,      SETALL, GETALL, SETVAL
SEM_ARRAY_VAL        DS    FL2    semaphore value
SEM_BUF_ELE          DSECT ,      sembuf element - semop
SEM_NUM              DS    FL2    semaphore number (0 to n-1)
SEM_OP               DS    FL2    semaphore operation
SEM_FLG              DS    H      operation flags
SEM#BUFLEN           EQU   *-SEM_BUF_ELE
* Flag bits - semop (also IPC_NOWAIT
SEM_UNDO             EQU   2      Set up adjust on exit entry.
* Control Commands - (also IPC_RMID, IPC_SET, IPC_STAT):
```

```
SEM_GETVAL           EQU   21     Get the current semaphore value
SEM_SETVAL           EQU   22     Change the semaphore value
SEM_GETPID           EQU   23     Get PID last process to alter sem
SEM_GETNCNT          EQU   24     Get count tasks waiting for val>0
SEM_GETZCNT          EQU   25     Get count tasks waiting for val=0
SEM_GETALL           EQU   26     Get the current semaphore values
SEM_SETALL           EQU   27     Change the semaphore values
* Maximum and minimum values
SEM#MAX_VAL          EQU   32767  Maximum sem_val (min = 0)
SEM#MAX_ADJ          EQU   16383  Maximum sem_adj (min = -MAX)
** BPXYSEM End
```

BPXYSHM — Map Interprocess Communications Shared Memory Segments



Purpose

Use the BPXYSHM macro to map the data structure and constants for the OpenExtensionscallable services that create and control interprocess communications shared memory segments.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

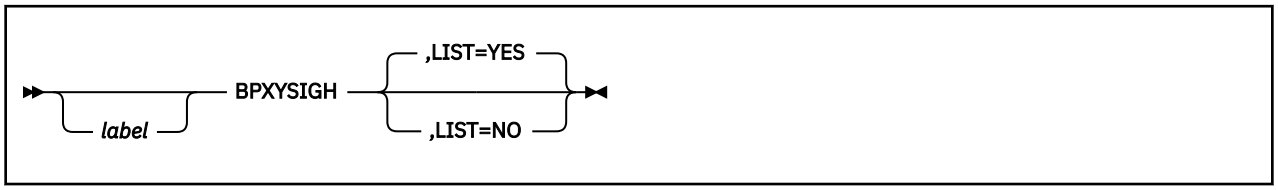
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYSHM macro expands as follows:

```
          BPXYSHM    ,
** BPXYSHM: Interprocess Communications Permission
**  Used By: XMC
SHMID_DS              DSECT ,      SHMID_DS - shmctl structure
SHM_PERM              DS    CL(IPC#LENGTH)   Mapped by BPXYIPCP
SHM_SEGSZ             DS    F      size of segment in bytes
SHM_LPID              DS    F      process ID of last operation
SHM_CPID              DS    F      process ID of creator
SHM_NATTCH            DS    F      number of current attaches
SHM_ATIME             DS    F      time of last shmat
SHM_DTIME             DS    F      time of last shmdt
SHM_CTIME             DS    F      time of last change shmget/shmctl
* Mode bits (mapped over S_TYPE in BPXYMODE):
SHM_RDONLY            EQU   1      Attach read-only (else read-write)
SHM_RND               EQU   2      Round attach address to SHMLBA
SHMLBA                EQU   4096   Rounding boundary
SHM#LENGTH            EQU   *-SHMID_DS   Length of this DSECT
** BPXYSHM End
```

BPXYSIGH — Map Signal Constants



Purpose

Use the BPXYSIGH to map the signal constants used by OpenExtensions callable services.

Parameters

label
is an optional assembler label for the statement.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

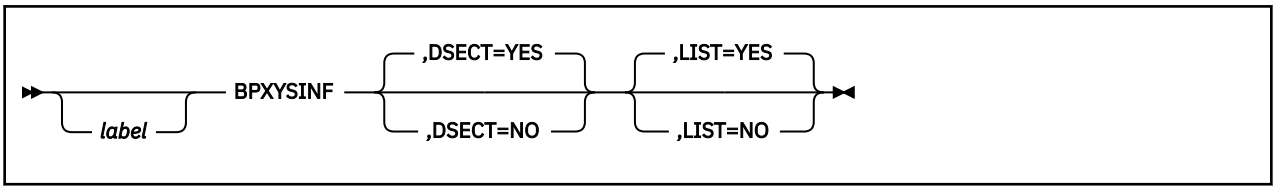
Usage Notes

1. The DSECT= parameter is allowed but ignored.
2. The PRINT OFF assembler statement overrides LIST=YES.
3. The BPXYSIGH mapping macro expands as follows:

```
         BPXYSIGH   ,
*********************************************************************
* Signals with default action ABNORMAL TERMINATION
SIGHUP#   EQU   1   Hangup detected on controlling terminal
SIGINT#   EQU   2   Interactive attention
SIGABRT#  EQU   3   Abnormal termination
SIGILL#   EQU   4   Detection of an incorrect hardware instruction
SIGFPE#   EQU   8   Erroneous arithmetic operation, such as division
*                   by zero of an operation resulting in overflows
SIGKILL#  EQU   9   Termination (cannot be caught or ignored)
SIGSEGV#  EQU  11   Detection of an incorrect memory reference
SIGPIPE#  EQU  13   Write on a pipe with no readers
SIGALRM#  EQU  14   Timeout
SIGTERM#  EQU  15   Termination
SIGUSR1#  EQU  16   Reserved as application-defined signal 1
SIGUSR2#  EQU  17   Reserved as application-defined signal 2
SIGABND#  EQU  18   Abend
SIGQUIT#  EQU  24   Interactive termination
SIGTRAP#  EQU  26   Reserved
* Signals with default action IGNORE THE SIGNAL
SIGNULL#  EQU   0   Null - no signal sent
SIGCHLD#  EQU  20   Child process terminated or stopped
SIGIO#    EQU  23   Completion of input or output
* Signals with default action STOP
SIGSTOP#  EQU   7   Stop (cannot be caught or ignored)
SIGTTIN#  EQU  21   Read from a control terminal attempted by a
*                   member of a background process group
SIGTTOU#  EQU  22   Write from a control terminal attempted by a
*                   member of a background process group
SIGTSTP#  EQU  25   Interactive stop
* Signals with default action CONTINUE IF IT IS CURRENTLY STOPPED,
*                             OTHERWISE IGNORE THE SIGNAL
SIGCONT#  EQU  19   Continue if stopped
*********************************************************************
** Equates that define sa_handler values on Sigaction
*********************************************************************
SIG_DFL#  EQU   0   Default signal action
SIG_IGN#  EQU   1   Ignore signal action
```

```
*********************************************************************
** Constants that define sa_flags values on Sigaction
*********************************************************************
SA_FLAGS_DFT#  EQU X'00000000'  Default sa_flags
SA_NOCLDSTOP#  EQU X'80000000'  No SIGCHLD when children stop
SA_OLD_STYLE#  EQU X'40000000'  Old style signal function
*********************************************************************
** Constants that define How parameter on sigprocmask
*********************************************************************
SIG_BLOCK#      EQU    0  Block signals set on in New_signal_mask
SIG_UNBLOCK#    EQU    1  Unblock signals set on in New_signal_mask
SIG_SETMASK#    EQU    2  Set signal mask to New_signal_mask
```

BPXYSINF — Map the Siginfo_t Structure for the wait-extensions Service



Purpose

Use the BPXYSINF macro to map the Siginfo_t structure used by the wait-extensions (BPX1WTE) callable service.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

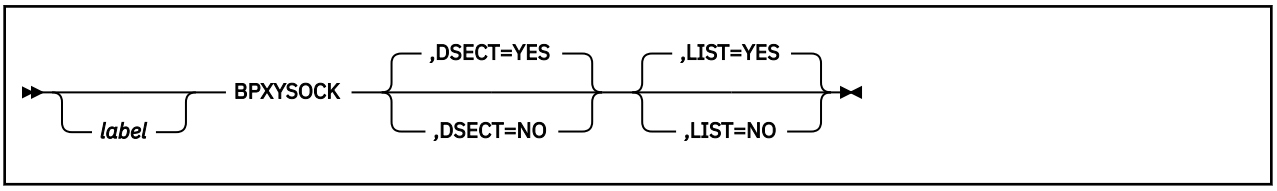
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYSINF macro expands as follows:

```
          BPXYSINF   ,
SIGINFO_T            DSECT ,      Siginfo_t structure
SI_SIGNO             DS    F      signal number
SI_ERRNO             DS    F      error number
SI_CODE              DS    F      signal code
SI_PID               DS    F      sending process ID
SI_UID               DS    F      real user ID of sending process
SI_ADDR              DS    A      address of faulting instruction
SI_STATUS            DS    F      exit value or signal
SI_BAND              DS    F      band event for SIGPOLL
SIGINFO#LENGTH       EQU *-SIGINFO_T Length of this DSECT
```

BPXYSOCK — Map the SOCKADDR Structure and Constants for Socket-Related Services



Purpose

Use the BPXYSOCK macro to map the SOCKADDR data structure and constants used by the socket-related OpenExtensions callable services.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYSOCK macro expands as follows:

```
            BPXYSOCK   ,
*
*********************************************************************
** BPXYSOCK: OpenVM Socket Address Structure                        *
** Used By: Sockets PFS                                             *
*********************************************************************
*
SOCKADDR             DSECT ,
SOCKBEGIN            DS    0F
*
SOCK_LEN             DS    X          Address Length - Length of      *X
                                      either SOCK_SIN (for AF_INET    *X
                                      sockets) or of the name supplied*X
                                      in SOCK_SUN_NAME (for AF_UNIX   *X
                                      sockets)
SOCK_FAMILY          DS    X          Address Family
SOCK_DATA            DS    0C         Protocol specific area
*
SOCK#LEN             EQU   *-SOCKADDR Constant - Fixed length of SOCK
*
*********************************************************************
*                                                                   *
*   AF_Inet Socket Address Structure                                *
*                                                                   *
```

```
*********************************************************************
*
                    ORG  SOCK_DATA   Start of AF_Inet unique area
SOCK_SIN            DS   0C
SOCK_SIN_PORT       DS   H           Port number used by the appl
SOCK_SIN_ADDR       DS   CL4         INET address (netid)
                    DS   CL8         Reserved area not used
*
SOCK_SIN#LEN        EQU  *-SOCK_SIN  Constant - Fixed length of
*                                    AF_Inet unique area
*
*
*********************************************************************
*                                                                   *
*   AF_Unix Socket Address Structure                                *
*                                                                   *
*********************************************************************
*
                    ORG  SOCK_DATA   Start of AF_Unix unique area
SOCK_SUN            DS   0C
SOCK_SUN_NAME       DS   CL108       Path name of the socket
*                                    Length 108 matches RS/6000
*
SOCK_SUN#LEN        EQU  *-SOCK_SUN  Constant - Fixed length of
*                                    AF_Unix unique area
*
*********************************************************************
*                                                                   *
*   AF_Inet6 Socket Address Structure                               *
*                                                                   *
*********************************************************************
*
                    ORG  SOCK_DATA   Start of AF_Inet6 area
SOCK_SIN6           DS   0C
SOCK_SIN6_PORT      DS   H           Port number used by the appl
SOCK_SIN6_FLOWINFO  DS   CL4         FLOW INFORMATION
SOCK_SIN6_ADDR      DS   CL16        INET address (netid)
SOCK_SIN6_SCOPE_ID  DS   CL4         SCOPE ID
*
SOCK_SIN6#LEN       EQU  *-SOCK_SIN6 Length of AF_INET6 area
*
*********************************************************************
*                                                                   *
*   Equates for Address Families                                    *
*                                                                   *
*********************************************************************
*
AF_UNSPEC           EQU  0           Unspecified
AF_UNIX             EQU  1           Unix Domain
AF_INET             EQU  2           Internetwork: UDP TCP
AF_IMPLINK          EQU  3           Arpanet imp addresses
AF_PUP              EQU  4           pup protocols: BSP
AF_CHAOS            EQU  5           mit CHAOS protocols
AF_NS               EQU  6           XEROX NS protocols
AF_NBS              EQU  7           nbs protocols
AF_ECMA             EQU  8           European computer man.
AF_DATAKIT          EQU  9           datakit protocols
AF_CCITT            EQU  10          CCITT protocols: X.25
AF_SNA              EQU  11          IBM SNA
AF_DECNET           EQU  12          DECNet
AF_DLI              EQU  13          Direct data link interface
AF_LAT              EQU  14          LAT
AF_HYLINK           EQU  15          NSC hyperchannel
AF_APPLETALK        EQU  16          Apple Talk
AF_IUCV             EQU  17          IBM IUCV
AF_ESCON            EQU  18          ESCON UDP
AF_INET6            EQU  19          IPv6
AF_ROUTE            EQU  20          Routing Sockets
AF_MAX              EQU  21
*
*********************************************************************/
*   Equates for protocol
*********************************************************************/
*
IPPROTO_IP          EQU   0          DEFAULT PROTOCOL
IPPROTO_TCP         EQU   6          TCP
IPPROTO_UDP         EQU  17          USER DATAGRAM
IPPROTO_IPV6        EQU  41          IPv6
IPPROTO_ICMPV6      EQU  58          IPv6 ICMP
*
IPPROTO_HOPOPTS     EQU   0
IPPROTO_ROUTING     EQU  43
```

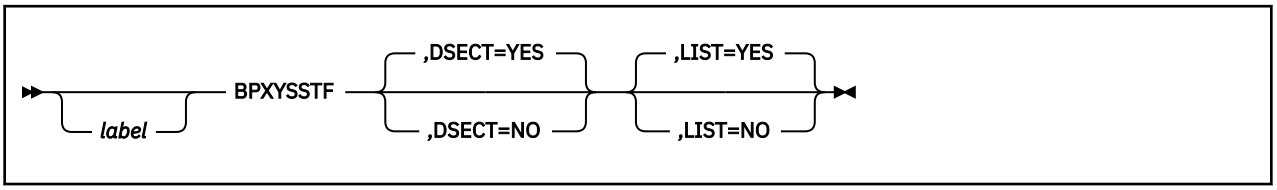
```
IPPROTO_FRAGMENT     EQU  44
IPPROTO_ESP          EQU  50
IPPROTO_AH           EQU  51
IPPROTO_NONE         EQU  59
IPPROTO_DSTOPTS      EQU  60
*
*********************************************************************
*                                                                   *
*   Equates for setpeer options                                     *
*                                                                   *
*********************************************************************
*
SOCK#SO_SET          DC   X'00000200'
SOCK#SO_UNSET        DC   X'00000400'
*
*********************************************************************
*                                                                   *
*   Equates for socket types                                        *
*                                                                   *
*********************************************************************
*
SOCK#_STREAM         EQU  1
SOCK#_DGRAM          EQU  2
SOCK#_RAW            EQU  3
SOCK#_RDM            EQU  4
SOCK#_SEQPACKET      EQU  5
*
*********************************************************************
*                                                                   *
*   Equates for Dimension   (socket syscall)                        *
*                                                                   *
*********************************************************************
*
SOCK#DIM_SOCKET      EQU  1
SOCK#DIM_SOCKETPAIR  EQU  2
*
*********************************************************************
*                                                                   *
*   Equates for getname option                                      *
*                                                                   *
*********************************************************************
*
SOCK#GNMOPTGETPEERNAME EQU  1
SOCK#GNMOPTGETSOCKNAME EQU  2
*
*********************************************************************
*                                                                   *
*   Equates for sockopt                                             *
*                                                                   *
*********************************************************************
*
SOCK#OPTOPTGETSOCKOPT    EQU  1
SOCK#OPTOPTSETSOCKOPT    EQU  2
SOCK#OPTOPTSETIBMSOCKOPT EQU  3
*
*********************************************************************
*                                                                   *
*   Equates for Shutdown options                                    *
*                                                                   *
*********************************************************************
*
SOCK#SHUTDOWNREAD      EQU  0
SOCK#SHUTDOWNWRITE     EQU  1
SOCK#SHUTDOWNBOTH      EQU  2
*
*
*********************************************************************
*                                                                   *
*   Equate for Level Number for socket options                      *
*                                                                   *
*********************************************************************
*
SOCK#SOL_SOCKET      DC   X'0000FFFF'
*
*
*********************************************************************
*                                                                   *
*   Equate for InAddrAny for bind requests                          *
*                                                                   *
*********************************************************************
*
INADDR_ANY           DC   X'00000000'
```

```
*
INADDR_LOOPBACK      DC   X'7F000001'
IN6ADDR_ANY          DC   X'00000000000000000000000000000000'
IN6ADDR_LOOPBACK     DC   X'00000000000000000000000000000001'
IN6ADDR_MAPPEDV4     DC   X'00000000000000000000FFFF'
IN6ADDR_COMPATV4     DC   X'000000000000000000000000'
*
*********************************************************************
*                                                                   *
*   Equates for Socket options                                      *
*                                                                   *
*********************************************************************
*
SOCK#SO_DEBUG        DC   X'00000001'
SOCK#SO_ACCEPTCONN   DC   X'00000002'
SOCK#SO_REUSEADDR    DC   X'00000004'
SOCK#SO_KEEPALIVE    DC   X'00000008'
SOCK#SO_DONTROUTE    DC   X'00000010'
SOCK#SO_BROADCAST    DC   X'00000020'
SOCK#SO_USELOOPBACK  DC   X'00000040'
SOCK#SO_LINGER       DC   X'00000080'
SOCK#SO_OOBINLINE    DC   X'00000100'
*
SOCK#SO_SNDBUF       DC   X'00001001'
SOCK#SO_RCVBUF       DC   X'00001002'
SOCK#SO_SNDLOWAT     DC   X'00001003'
SOCK#SO_RCVLOWAT     DC   X'00001004'
SOCK#SO_SNDTIMEO     DC   X'00001005'
SOCK#SO_RCVTIMEO     DC   X'00001006'
SOCK#SO_ERROR        DC   X'00001007'
SOCK#SO_TYPE         DC   X'00001008'
*
* Non-standard sockopts
*
SO_PROPAGATEID       DC   X'00004000'                        /*
SO_CLUSTERCONNTYPE   DC   X'00004001'
SO_SECINFO           DC   X'00004002'
*
* SO_CLUSTERCONNTYPE Output Values
*
SO_CLUSTERCONNTYPE_NOCONN        EQU  0
SO_CLUSTERCONNTYPE_NONE          EQU  1
SO_CLUSTERCONNTYPE_SAME_CLUSTER  EQU  2
SO_CLUSTERCONNTYPE_SAME_IMAGE    EQU  4
SO_CLUSTERCONNTYPE_INTERNAL      EQU  8
*
*
* IPPROTO_IP Options
*
IP_TOS                    EQU 2                              /*
IP_MULTICAST_TTL          EQU 3                              /*
IP_MULTICAST_LOOP         EQU 4                              /*
IP_ADD_MEMBERSHIP         EQU 5                              /*
IP_DROP_MEMBERSHIP        EQU 6                              /*
IP_MULTICAST_IF           EQU 7                              /*
IP_DEFAULT_MULTICAST_TTL  EQU 1                              /*
IP_DEFAULT_MULTICAST_LOOP EQU 1                              /*
IP_MAX_MEMBERSHIPS        EQU 20                             /*
*
* setibmsockopt options
*
SOCK#SO_BULKMODE           DC X'00008000'
SOCK#SO_IGNOREINCOMINGPUSH DC X'00000001'
SOCK#SO_NONBLOCKLOCAL      DC X'00008001'
SOCK#SO_IGNORESOURCEVIPA   DC X'00000002'
*                       Toggles the use of non-VIPA addresses.  When
*                       enabled, non-VIPA addresses will be used for
*                       outbound IP packets.
SOCK#SO_OPTMSS             DC X'00008003'
*                Toggles the use of optimal TCP segment size.
*                When enabled, the TCP segment size may be optimally
*                increased on outbound data transfers.  This may
*                reduce the amount of TCP outbound and inbound
*                acknowledgement packet processing; therefore,
*                minimizing CPU consumption.
SOCK#SO_OPTACK             DC X'00008004'   Optimize Acks
SOCK#SO_EIOIFNEWTP         DC X'00000005'   Notify of new tp
*********************************************************************
*                                                                   *
*   Equates for So_   option values                                 *
*                                                                   *
*********************************************************************
```

```
SOCK#SO_SETOPTIONON   DC X'00000001'
SOCK#SO_SETOPTIONOFF  DC X'00000000'
*********************************************************************
*                                                                   *
*   Equates for IPPROTO_TCP options                                 *
*                                                                   *
*********************************************************************
SOCK#TCP_NODELAY      DC X'00000001'
SOCK#TCP_KEEPALIVE    DC X'00000008'
*
*********************************************************************
*                                                                   *
*   Equates for Socket Port Constant                                *
*                                                                   *
*********************************************************************
*
SOCK#LASTRESERVEPORT   EQU  1023
*
*
IP_MREQ         DSECT ,
IMR_MULTIADDR    DS CL4            IP MULTICAST ADDR OF GROUP
IMR_INTERFACE    DS CL4            LOCAL IP ADDR OF INTERFACE
*
*********************************************************************
*                                                                   *
*   Structure for So_Linger                                         *
*
*********************************************************************
*
SOCK_LINGER_STRUCT DSECT ,
SOCK_L_ONOFF         DS    F          On/Off indicator
SOCK_L_LINGER        DS    F          Length of time to linger
*********************************************************************
*                                                                   *
*   Equates for IPPROTO_IPV6 Options
*                                                                   *
*********************************************************************
SOCK#IPV6_UNICAST_HOPS    EQU 3
SOCK#IPV6_MULTICAST_LOOP  EQU 4
SOCK#IPV6_JOIN_GROUP      EQU 5
SOCK#IPV6_LEAVE_GROUP     EQU 6
SOCK#IPV6_MULTICAST_IF    EQU 7
SOCK#IPV6_MULTICAST_HOPS  EQU 9
SOCK#IPV6_V6ONLY          EQU 10
SOCK#IPV6_HOPLIMIT        EQU 11         /* ANC DATA ONLY  */
SOCK#IPV6_PKTINFO         EQU 13
SOCK#IPV6_RECVHOPLIMIT    EQU 14
SOCK#IPV6_RECVPKTINFO     EQU 15
SOCK#IPV6_REACHCONF       EQU 17
SOCK#IPV6_USE_MIN_MTU     EQU 18
SOCK#IPV6_CHECKSUM        EQU 19
*********************************************************************
*   The following are not currently supported by TCPIP
*********************************************************************
SOCK#IPV6_PATHMTU          EQU 12
SOCK#IPV6_RECVPATHMTU      EQU 16
SOCK#IPV6_NEXTHOP          EQU 20
SOCK#IPV6_RTHDR            EQU 21
SOCK#IPV6_HOPOPTS          EQU 22
SOCK#IPV6_DSTOPTS          EQU 23
SOCK#IPV6_RTHDRDSTOPTS     EQU 24
SOCK#IPV6_RECVRTHDR        EQU 25
SOCK#IPV6_RECVHOPOPTS      EQU 26
SOCK#IPV6_RECVRTHDRDSOPTS  EQU 27
SOCK#IPV6_RECVDSTOPTS      EQU 28
SOCK#IPV6_RTHDR_TYPE_0     EQU 0        IPv6 Routing hdr type 0
*********************************************************************
*                                                                   *
*   Equates for IPPROTO_ICMPV6 options
*                                                                   *
*********************************************************************
SOCK#ICMP6_FILTER          EQU 1
*********************************************************************
*                                                                   *
*   Structure for Packet Source/Destination Information
*                                                                   *
*********************************************************************
*
IN6_PKTINFO        DSECT ,
IPI6_ADDR            DS    CL16       IPv6 Addr
IPI6_IFINDEX         DS    F          Interface Index
*********************************************************************
```

```
*                                                                     *
*   Structure for Multicast Mreq
*                                                                     *
***********************************************************************
*
IPV6_MREQ          DSECT ,
IPV6MR_MULTIADDR     DS    CL16       IPv6 Addr
IPV6MR_INTERFACE     DS    F          Interface index
***********************************************************************
*                                                                     *
*   Structure for CInet Interface Index
*                                                                     *
***********************************************************************
*
IFINDEX            DSECT ,
IFI_TDX              DS    H          Cinet Td Index
IFI_INDEX            DS    H          Stacks  Interface Index
***********************************************************************
*                                                                     *
*   Structure for Icmp6 Filtering
*                                                                     *
***********************************************************************
*
ICMP6_FILTER       DSECT ,
ICMP6_FILT           DS    8F         8*32 = 256 bits
*
** BPXYSOCK End
```

BPXYSSTF — Map the File System Status Structure



Purpose

Use the BPXYSSTF macro to map the file system status structure returned by the fstatvfs (BPX1FTV), statvfs (BPX1STV), and w_statvfs (BPX1STF) callable services.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

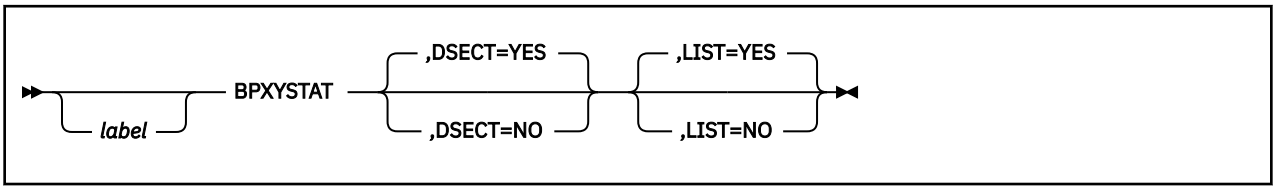
Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYSSTF mapping macro expands as follows:

```
              BPXYSSTF   ,
** BPXYSSTF: file system status response structure
**  Used By: STF STV FTV VSF
SSTF                  DSECT ,
SSTFID                DC    C'SSTF'  EBCDIC ID - SSTF (f_OEcbid)
SSTFLEN               DC    A(SSTF#LENGTH)  Length of SSTF (f_OEcblen
SSTFBLOCKSIZE         DS    F       Block size (f_bsize)
                      DS    F       Reserved
SSTFDBLTOTSPACE       DS    0D      Name of dblword field - total
                      DS    F       Reserved
SSTFTOTALSPACE        DS    F       Total space. The total number of
                                    blocks on file system in units of
                                    f_frsize (f_blocks)
SSTFDBLUSEDSPACE      DS    0D      Name of dblword field - used
                      DS    F       Reserved
SSTFUSEDSPACE         DS    F       Allocated space in block size unit
                                    (f_OEusedspace)
SSTFDBLFREESPACE      DS    0D      Name of dblword field - free
                      DS    F       Reserved
SSTFFREESPACE         DS    F       Space available to unprivileged
                                    users in block size units
                                    (f_bavail)
SSTFENDVER1           EQU   *       End of Version 1 SSTF
SSTFFSID              DS    F       File system ID (f_fsid)
                                    Set by LFS
SSTFFLAG              DS    0BL.32  Bit mask of f_flag vals
SSTFFLAGB1            DS    XL1     byte 1
```

```
SSTFEXPORTED          EQU   X'40'  Filesys is exported
                                   (ST_OEEXPORTED)
                                   Set by LFS
SSTFV3PROP            DS    XL1    NFS V3 Properties
SSTFFSF_V3RET         EQU   X'80'   V3 Prop Returned
SSTFFSF_CANSETTIME    EQU   X'10'   time_delta accuracy
SSTFFSF_HOMOGENEOUS   EQU   X'08'   Pathconf same for all
SSTFFSF_SYMLINK       EQU   X'02'   Supports Symlinks
SSTFFSF_LINK          EQU   X'01'   Supports Hard Links
SSTFFLAGB3            DS    XL1    byte 3
SSTFFLAGB4            DS    XL1    byte 4
SSTFNOSEC             EQU   X'04'  No Security checks enforced
SSTFNOSUID            EQU   X'02'  SetUID/SetGID not supported
                                   (ST_NOSUID)
                                   Set by LFS
SSTFRDONLY            EQU   X'01'  Filesys is read only
                                   (ST_RDONLY)
                                   Set by LFS
SSTFMAXFILESIZE       DS    0D     Name of dblword field - maximum
                                   file size
                                   May be set by LFS
SSTFMAXFILESIZEHW     DS    F      High word of max file size
                                   (f_OEmaxfilesizehw)
SSTFMAXFILESIZELW     DS    F      Low word of max file size
                                   (f_OEmaxfilesizelw)
                      DS    CL16   Reserved
SSTFENDLFSINFO        EQU   *      End of LFS information
SSTFFRSIZE            DS    F      Fundamental filesystem block size
                                   (f_frsize)
                      DS    F      Reserved
SSTFDBLBFREE          DS    0D     Name of dblword field -
                                   total number of free blocks
                      DS    F      Reserved
SSTFBFREE             DS    F      Total number of free blocks
                                   (f_bfree)
SSTFFILENODES         DS    0CL12  File nodes
SSTFFILES             DS    F      Total number of file nodes
                                   in the file system (f_files)
SSTFFFREE             DS    F      Total number of free file nodes
                                   (f_ffree)
SSTFFAVAIL            DS    F      Number of free file nodes available
                                   to unprivileged users (f_favail)
SSTFNAMEMAX           DS    F      Maximum file name len (f_namemax)
SSTFINVARSEC          DS    F      Number of seconds file system
                                   will remain unchanged
                                   (f_OEinvarsec)
SSTFTIME_DELTA        DS    0CL8   Set file time granularity
SSTFTIME_DELTA_SEC    DS    F        Seconds
SSTFTIME_DELTA_NS     DS    F        Nano-seconds
                      DS    CL12   Reserved
SSTF#LENGTH           EQU   *-SSTF Length of this structure
SSTF#MINLEN           EQU   SSTFENDVER1-SSTF
SSTF#LFSLEN           EQU   SSTFENDLFSINFO-SSTF
** BPXYSSTF End
```

BPXYSTAT — Map the File Status Structure for the stat Service



Purpose

Use the BPXYSTAT macro to map the file status structure returned by the stat (BPX1STA) callable service.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

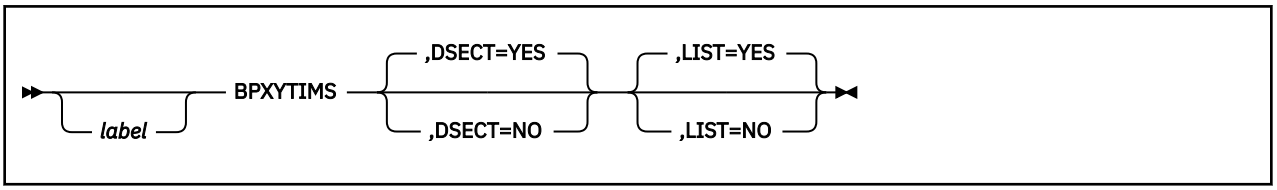
Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYSTAT mapping macro expands as follows:

```
         BPXYSTAT   ,
STAT                DSECT ,
ST_BEGIN            DS    0D
*
ST_EYE              DC    C'STAT'  Eye-catcher
ST_LENGTH           DC    AL2(STAT#LENGTH)                          X
                          Length of this structure
ST_VERSION          DC    AL2(ST#VER)                               X
                          Version of this structure
ST_MODE             DS    F       File Mode, mapped by BPXYMODE
ST_INO              DS    F       File Serial Number
ST_DEV              DS    F       Device ID of the file
ST_NLINK            DS    F       Number of links
ST_UID              DS    F       User ID of the owner of the file
ST_GID              DS    F       Group ID of the Group of the file
ST_SIZE             DS    0D      File Size in bytes, for regular
*                                 files.  Unspecified, for others
ST_SIZE_H           DS    F       First word of size
ST_SIZE_L           DS    F       Second word of size
ST_ATIME            DS    F       Time of last access
ST_MTIME            DS    F       Time of last data modification
ST_CTIME            DS    F       Time of last file status change
*                                 Time is in seconds since
*                                 00:00:00 GMT, Jan. 1, 1970
ST_RDEV             DS    0F      Device Information
ST_MAJORNUMBER      DS    H       Major number for this file, if it
*                                    is a character special file.
ST_MINORNUMBER      DS    H       Minor number for this file, if it
```

```
*                                    is a character special file.
ST_AUDITORAUDIT      DS    F      Area for auditor audit info
ST_USERAUDIT         DS    F      Area for user audit info
ST_BLKSIZE           DS    F      File Block size
ST_CREATETIME        DS    F      File Creation Time
ST_AUDITID           DS    4F     RACF File ID for auditing
ST_RES01             DS    F
ST_CHARSETID         DS    3F     Coded Character Set ID
ST_BLOCKS_D          DS    0D     Double word number - blocks allocated
ST_RES02             DS    F
ST_BLOCKS            DS    F      Number of blocks allocated
ST_RES03             DS    9F     Area for future expansion
*
*   Constants
*
ST#VER               EQU   ST#VER01 Current version
ST#VER01             EQU   1        Version 1 of this structure
STAT#LENGTH          EQU   *-STAT   Length of STAT
ST#LEN               EQU   STAT#LENGTH Length of STAT
```

BPXYTIMS — Map the Processor Time Structure for the times Service



Purpose

Use the BPXYTIMS macro to map the processor time structure returned by the times (BPX1TIM) callable service.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

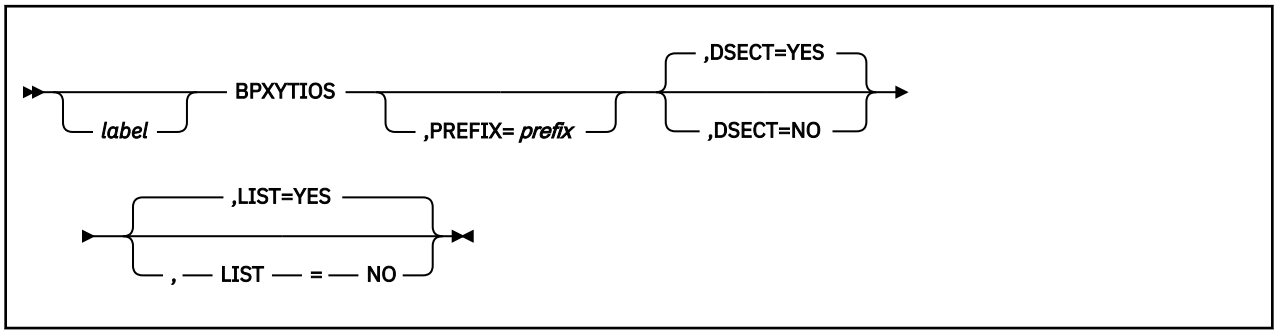
Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYTIMS mapping macro expands as follows:

```
         BPXYTIMS   ,
TIMS                DSECT ,
TIMSBEGIN           DS    0F
TIMSUTIME           DS    F       User CPU time of current process
*                                 in hundredths of a second.
*                                 This includes time spent in the
*                                 user CMS process.
TIMSSTIME           DS    F       System CPU time of current process
*                                 in hundredths of a second.
*                                 This includes the time spent in
*                                 the root CMS process
TIMSCUTIME          DS    F       Sum of user CPU time values (as
*                                 defined in TIMSUTIME) and child user
*                                 CPU time values (as defined in
*                                 TIMSCUTIME) for all waited-for
*                                 child processes.  Zero if the
*                                 current process has no waited-for
*                                 children.
TIMSCSTIME          DS    F       Sum of system CPU time values (as
*                                 defined in TIMSSTIME) and child
*                                 system CPU time values (as defined in
*                                 TIMSCSTIME) for all waited-for
*                                 child processes.  Zero if the
*                                 current process has no waited-for
```

```
*                                  children.
TIMS#LENGTH          EQU   *-TIMS Length of this structure
```

BPXYTIOS — Map the termios Structure



Purpose

Use the BPXYTIOS macro to map the termios structure used by OpenExtensions callable services.

Parameters

label
is an optional assembler label for the statement.

PREFIX=*prefix*
makes the labels unique. The characters specified on this parameter will be appended before each label.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

Usage Notes

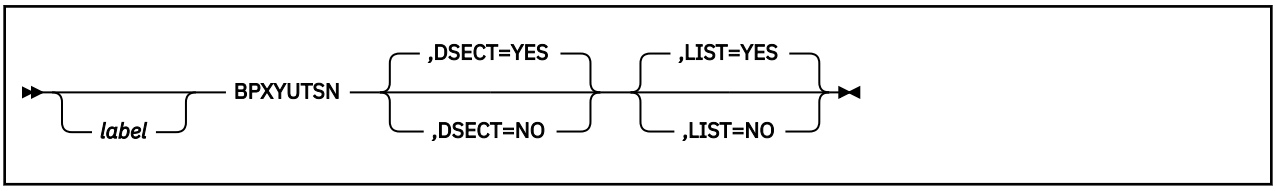
1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYTIOS macro expands as follows:

```
         BPXYTIOS   , PREFIX=
BPXYTIOS DSECT ,                  Define DSECT
* baud rate values
B0          EQU    0              0     baud (hang-up)
B50         EQU    1              50    baud
B75         EQU    2              75    baud
B110        EQU    3              110   baud
B134        EQU    4              134.5 baud
B150        EQU    5              150   baud
B200        EQU    6              200   baud
B300        EQU    7              300   baud
B600        EQU    8              600   baud
B1200       EQU    9              1200  baud
B1800       EQU   10              1800  baud
B2400       EQU   11              2400  baud
B4800       EQU   12              4800  baud
B9600       EQU   13              9600  baud
```

```
B19200      EQU   14              19200 baud
B38400      EQU   15              38400 baud
* Values for c_cflag field are bitwise distinct except for
* character size bits, which form a number.
CLOCAL      EQU   X'01'           Ignore modem status lines
CREAD       EQU   X'02'           Enable receiver
CSIZE       EQU   X'30'           Character size bits
CS5         EQU   X'00'             B'00' - 5 bits/character
CS6         EQU   X'10'             B'01' - 6 bits/character
CS7         EQU   X'20'             B'10' - 7 bits/character
CS8         EQU   X'30'             B'11' - 8 bits/character
CSTOPB      EQU   X'80'           Send two stop bits, else one
HUPCL       EQU   X'01'           Hang up on last close
PARENB      EQU   X'02'           Parity enable
PARODD      EQU   X'04'           Odd parity, else even
* c_cflag offsets for bits defined above.  These values are
* used to refer to the correct byte within c_cflag.  For
* instance, "TM    C_CFLAG+HUPCL_O,HUPCL".
CLOCAL_O    EQU   3
CREAD_O     EQU   3
CSIZE_O     EQU   3
CS5_O       EQU   3
CS6_O       EQU   3
CS7_O       EQU   3
CS8_O       EQU   3
CSTOPB_O    EQU   3
HUPCL_O     EQU   2
PARENB_O    EQU   2
PARODD_O    EQU   2
* Values for c_lflag field are bitwise-distinct.
ECHO        EQU  X'01'            Enable echo
ECHOE       EQU  X'02'            Echo ERASE as error correcting      X
                                    backspace
ECHOK       EQU  X'04'            Echo KILL
ECHONL      EQU  X'08'            Echo new line
ICANON      EQU  X'10'            Canonical input
IEXTEN      EQU  X'20'            Enable extended functions
ISIG        EQU  X'40'            Enable signals
NOFLSH      EQU  X'80'            Disable flush after interrupt,      X
                                    quit, or suspend
TOSTOP      EQU  X'01'            Send SIGTTOU for background         X
                                    output
* c_lflag offsets for bits defined above.  These values are
* used to refer to the correct byte within c_lflag.  For
* instance, "TM    C_LFLAG+TOSTOP_O,TOSTOP".
ECHO_O      EQU  3
ECHOE_O     EQU  3
ECHOK_O     EQU  3
ECHONL_O    EQU  3
ICANON_O    EQU  3
IEXTEN_O    EQU  3
ISIG_O      EQU  3
NOFLSH_O    EQU  3
TOSTOP_O    EQU  2* Values for c_iflag field are bitwise-distinct.
BRKINT      EQU  X'01'            Signal interrupt on break
ICRNL       EQU  X'02'            Map CR to NL on input
IGNBRK      EQU  X'04'            Ignore break condition
IGNCR       EQU  X'08'            Ignore CR
IGNPAR      EQU  X'10'            Ignore characters with parity       X
                                    errors
INLCR       EQU  X'20'            Map NL to CR in input
INPCK       EQU  X'40'            Enable input parity check
ISTRIP      EQU  X'80'            Strip character
IXOFF       EQU  X'01'            Enable start/stop input             X
                                    control
IXON        EQU  X'02'            Enable start/stop output            X
                                    control
PARMRK      EQU  X'04'            Mark parity errors
* c_iflag offsets for bits defined above.  These values are
* used to refer to the correct byte within c_iflag.  For
* instance, "TM    C_IFLAG+BRKINT_O,BRKINT".
BRKINT_O    EQU  3
ICRNL_O     EQU  3
IGNBRK_O    EQU  3
IGNCR_O     EQU  3
IGNPAR_O    EQU  3
INLCR_O     EQU  3
INPCK_O     EQU  3
ISTRIP_O    EQU  3
IXOFF_O     EQU  2
IXON_O      EQU  2
PARMRK_O    EQU  2
```

```
* Values for c_oflag are bitwise distinct.
OPOST       EQU X'01'              Perform output processing
*
* c_oflag offsets for bits defined above.  These values are
* used to refer to the correct byte within c_oflag.  For
* instance, "TM    C_OFLAG+OPOST_O,OPOST".
*
OPOST_O     EQU 3
* Optional actions used by tcsetattr
TCSANOW       EQU  0               Change occurs immediately
TCSADRAIN     EQU  1               Change occurs after all output      X
                                     has been written
TCSAFLUSH     EQU  2               Change occurs after all output      X
                                     has been written and input        X
                                     has been discarded
* queue selector values for tcflush
TCIFLUSH     EQU   0               Flush data received but not read
TCOFLUSH     EQU   1               Flush data written but not sent
TCIOFLUSH    EQU   2               Flush both data received but not    X
                                     read and data written but not sent
* action values for tcflow
TCOOFF      EQU 0                  Suspend output
TCOON       EQU 1                  Restart suspended output
TCIOFF      EQU 2                  Transmit STOP character
TCION       EQU 3                  Transmit START character
* Special Control Characters subscripts for cc_c
*   field
VINTR       EQU  0                 INTR  character
VQUIT       EQU  1                 QUIT  character
VERASE      EQU  2                 ERASE character
VKILL       EQU  3                 KILL  character
VEOF        EQU  4                 EOF   character
VEOL        EQU  5                 EOL   character
VMIN        EQU  6                 MIN   value
VSTART      EQU  7                 START character
VSTOP       EQU  8                 STOP  character
VSUSP       EQU  9                 SUSP  character
VTIME       EQU 10                 TIME  value
NCCS     EQU   11                  Number of special control chars
C_CFLAG    DC  F'0'                Control modes
C_IFLAG    DC  F'0'                Input modes
C_LFLAG    DC  F'0'                Local modes
C_OFLAG    DC  F'0'                Output modes
C_CC       DC  (NCCS)X'0'          Control characters and values
BPXYTIOS#LENGTH      EQU   *-BPXYTIOS Length of this structure
```

BPXYUTSN — Map the System Information Structure for the uname Service



Purpose

Use the BPXYUTSN macro to map the system information structure returned by the uname (BPX1UNA) callable service.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

Usage Notes

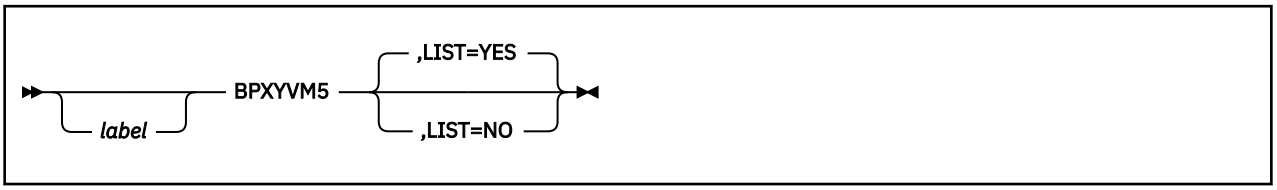
1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYUTSN mapping macro expands as shown below.
3. The formats of the UTSNAMERELEASE and UTSNAMEVERSION fields are described in Table 4 on page 481.

```
         BPXYUTSN   ,
UTSN                DSECT ,
UTSNAMESYSNAMELEN   DS    F      Length of UTSNAMESYSNAME string
UTSNAMESYSNAME      DS    CL16   Name of this implementation of the
*                                operating system (CMS)
UTSNAMENODENAMELEN  DS    F      Length of UTSNAMENODENAME string
UTSNAMENODENAME     DS    CL32   Name of this node within the
*                                communications network
UTSNAMERELEASELEN   DS    F      Length of UTSNAMERELEASE string
UTSNAMERELEASE      DS    CL64   Current CMS release level of this
*                                implementation
UTSNAMEVERSIONLEN   DS    F      Length of UTSNAMEVERSION string
UTSNAMEVERSION      DS    CL64   Current version level of this release
UTSNAMEMACHINELEN   DS    F      Length of UTSNAMEMACHINE string
UTSNAMEMACHINE      DS    CL16   Name of the hardware type on which
*                                the system is running
UTSN#LENGTH         EQU   *-UTSN Length of this structure
```

Table 4. Formats of the UTSNAMERELEASE and UTSNAMEVERSION Fields

Field	Description
UTSNAMERELEASE	The level of CMS in use, expressed as the string `CMS_`*`l`*`_`*`s`*`_`*`f`*, where: ***l*** is the CMS level as returned by QUERY CMSLEVEL. ***s*** is the 4-digit CMS service level as it appears in DMSLVLTB. ***f*** is the CMS level code returned by DMSQEFL in its output parameter *cms_level*. For example, the release (CMS) information for z/VM Version 3 Release 1.0 is: `CMS_16_0000_44`.
UTSNAMEVERSION	The level of CP in use, expressed as the string `CP_`*`v`*`.`*`r`*`.`*`m`*`_`*`s`*`_`*`f`*, where: ***v*** is the CP version number returned by QUERY CPLEVEL. ***r*** is the CP release number returned by QUERY CPLEVEL. ***m*** is the CP modification level returned by QUERY CPLEVEL. ***s*** is the 4-digit CP service level as it appears in the output of QUERY CPLEVEL. ***f*** is the CP level code returned by DMSQEFL in its output parameter *cp_level*. For example, the version (CP) information for z/VM Version 3 Release 1.0 is: `CP_3.1.0_0000_40`.

BPXYVM5 — Map Function Code Values for the openvmf Service



Purpose

Use the BPXYVM5 macro to map the function code values for the openvmf (BPX1VM5) callable service. BPXYVM5 consists only of equates.

Parameters

label
is an optional assembler label for the statement.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

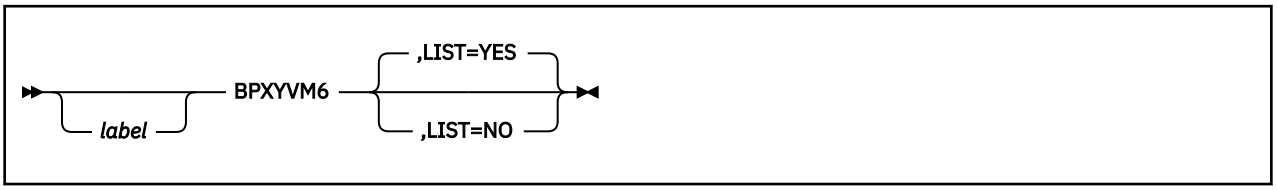
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYVM5 macro expands as follows:

```
         BPXYVM5    ,
*********************************************************************
* The following values can be used to set the function_code
* parameter.
*
VM5_RELEASE_TOKEN          EQU   1 Release BFS file tokens
VM5_FILEPOOL_ADMIN_RESPECT EQU   2 Respect file pool admin authority
VM5_FILEPOOL_ADMIN_IGNORE  EQU   3 Ignore file pool admin authority
VM5_RESOLVE_INO            EQU   4 Resolve INO into BFS path.
VM5_RESOLVE_PATH           EQU   5 Resolve path with links.
VM5_SET_SGID               EQU   6 Set supplementary GID.
VM5_SET_ALL_IDS            EQU   7 Set eUID, eGID, sGID.
VM5_GET_FILESYS_TYPE       EQU   8 Get file system type.
VM5_FSTYPE_BFS             EQU   C'BFS '
VM5_FSTYPE_CSI             EQU   C'CSI '
VM5_FSTYPE_PIP             EQU   C'PIP '
VM5_FSTYPE_SOC             EQU   C'SOC '
VM5_FSTYPE_NFS             EQU   C'NFS '
VM5_FSTYPE_LENGTH          EQU   4
```

BPXYVM6 — Map the Function Code Values for the setopen Service



Purpose

Use the BPXYVM6 macro to map the function code values for the setopen (BPX1VM6) callable service. BPXYVM6 consists only of equates.

Parameters

label
is an optional assembler label for the statement.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

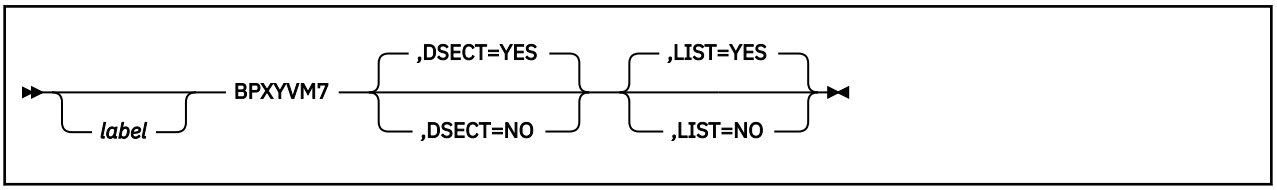
LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYVM6 macro expands as follows:

```
         BPXYVM6    ,
*********************************************************************
* The following values can be used to set the function_code
* parameter.
VM6_EXECLEVEL_OFF          EQU   0  Turn exec svc level off
VM6_EXECLEVEL_ON           EQU   1  Turn exec svc level on
```

BPXYVM7 — Map the Function Code Values and Buffer for the openvmf7 Service



Purpose

Use the BPXYVM7 macro to map the function code values and buffer contents for the openvmf7 (BPX1VM7) callable service.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

Usage Notes

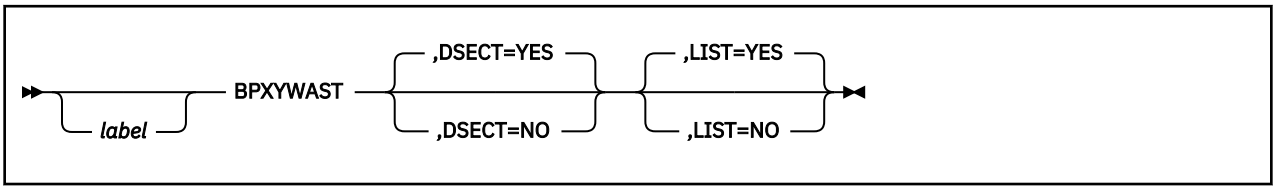
1. The BPXYVM7 mapping macro expands as follows:

```
VM7_VERSION1         EQU  1
VM7_GET_EXPORT_LIST  EQU  1
VM7_GET_DUMP_LIST    EQU  2
VM7_PCNFS_AUTH       EQU  3
VM7P_MAXGIDSIZE      EQU  16
VM7P_PCNFS_VERSION1  EQU  1
VM7P_PCNFS_VERSION2  EQU  2
VM7_MAXPATHLEN       EQU  1024
VM7_MAXNAMELEN       EQU  256
VM7_MAXCOMMENTLEN    EQU  256
VM7E_EXPORT_LIST     DS   0F
VM7E_VERSION         DS   F
VM7E_ENTRY_COUNT     DS   F
VM7E_ENTRY_TOTAL     DS   F
VM7E_ENTRY           DS   0F
VM7E_FILE_SYSTEM_LENGTH DS F
VM7E_WHO_COUNT       DS   F
VM7E_WHO_TOTAL       DS   F
VM7E_FILE_SYSTEM     DS   0C      * Variable length
VM7E_WHO_LIST        DS   0C
VM7E_WHO_LENGTH      DS   F
VM7E_WHO             DS   0C      * Variable length
VM7E_LENGTH          EQU  *-VM7E_EXPORT_LIST
VM7D_DUMP_LIST       DS   0F
VM7D_VERSION         DS   F
VM7D_ENTRY_COUNT     DS   F
VM7D_ENTRY_TOTAL     DS   F
```

```
VM7D_ENTRY           DS   0F
VM7D_HOST_LENGTH     DS   F
VM7D_HOST            DS   0C      * Variable length
VM7D_FILE_SYSTEM_LENGTH DS F
VM7D_FILE_SYSTEM     DS   0C      * Variable length
VM7D_LENGTH          EQU  *-VM7D_DUMP_LIST
VM7P_PCNFS_INPUT     DS   0F
VM7P_USERNAME_LENGTH DS   F
VM7P_USERNAME        DS   CL(VM7_MAXCOMMENTLEN)
VM7P_PASSWORD_LENGTH DS   F
VM7P_PASSWORD        DS   CL(VM7_MAXCOMMENTLEN)
VM7P_CMI_LENGTH      DS   F
VM7P_CMI             DS   CL(VM7_MAXCOMMENTLEN)
VM7P_INPUT_LENGTH    EQU  *-VM7P_PCNFS_INPUT
VM7P_PCNFS_OUTPUT    DS   0F
VM7P_VERSION         DS   F
VM7P_UID             DS   F
VM7P_GID             DS   F
VM7P_VERSION2_DATA   DS   0F
VM7P_GIDLIST_COUNT   DS   F
VM7P_GID_LIST        DS   CL(VM7P_MAXGIDSIZE*4)
VM7P_HOME_LENGTH     DS   F
VM7P_HOME            DS   CL(VM7_MAXPATHLEN)
VM7P_DEF_UMASK       DS   F
VM7P_CMO_LENGTH      DS   F
VM7P_CMO             DS   CL(VM7_MAXCOMMENTLEN)
VM7P_OUTPUT_LENGTH   EQU  *-VM7P_PCNFS_OUTPUT
```

2. In the VM7P_PCNFS_OUTPUT section, the information following VM7P_VERSION2_DATA is returned only when the *foreign_host* specified on the openvmf7 call supports Version 2 SUN PC-NFS requests (that is, when VM7P_VERSION is equal to VM7P_PCNFS_VERSION2).

BPXYWAST — Map the Wait Status Word



Purpose

Use the BPXYACC macro to map the wait status word used by OpenExtensions callable services.

Parameters

label
is an optional assembler label for the statement.

DSECT=YES
creates a DSECT for the macro. This is the default. Addressability requires a USING statement and a register pointing to storage.

DSECT=NO
allocates space for the macro in the current DSECT or CSECT. In a reentrant program, DSECT=NO places the macro in the current DSECT, and addressability is accomplished without the individual USING statement required by DSECT=YES. In a nonreentrant program, DSECT=NO places the macro in the current CSECT, and addressability is obtained through the program base registers.

LIST=YES
causes the expansion of the macro to appear in the listing. This is the default.

LIST=NO
removes the macro expansion from the listing.

Usage Notes

1. The PRINT OFF assembler statement overrides LIST=YES.
2. The BPXYWAST mapping macro expands as follows:

```
         BPXYWAST   ,
WAST                DSECT ,
                    DS    XL2    Reserved for IBM use - Set to zeros
WASTEXITSTATUS      DS    0XL2   Exit Status value passed on the
*                                BPX1EXI or BPX1MPC system calls
WASTEXITCODE        DS    0XL1   Exit return code for ending process
WASTSIGSTOP         DS    XL1    Signal that stopped process
WASTSIGTERM         DS    0XL1   Signal that terminated process
WASTSTOPFLAG        DS    XL1    Special flag value that qualifies the
*                                reason for the process being stopped.
* * WASTSTOPFLAG Values * * * * * * * * * * * * * * * * * * * * * * * *
WASTDUMP            EQU   X'80'  Bit 0 of WASTSTOPFLAG on, a core dump
*                                was taken when the process terminated
WASTSTOPFLAGSIG     EQU   X'7F'  Process stopped for a signal
WASTSTOPFLAGFORK    EQU   X'7E'  Process stopped for a fork
*                                (not currently supported)
WASTSTOPFLAGEXEC    EQU   X'7D'  Process stopped for an exec
*                                (not currently supported)
WAST#LENGTH         EQU   *-WAST Length of this structure
```

Appendix A. Return Codes

This appendix describes the return codes returned by OpenExtensions callable services. Two lists are provided. The first list is arranged by value and contains a description of each return code. The second list, in Table 6 on page 490, is arranged alphabetically and contains a cross-reference to the corresponding values.

OpenExtensions Return Codes Listed by Numeric Value

Table 5. OpenExtensions Return Codes by Numeric Value

Dec Value	Hex Value	Return Code	Description
1	0001	EDOM	Error in the domain
2	0002	ERANGE	Result is too large
111	006F	EACCES	Permission is denied
112	0070	EAGAIN	The resource is temporarily unavailable
113	0071	EBADF	The file descriptor is incorrect
114	0072	EBUSY	The resource is busy
115	0073	ECHILD	No child process exists
116	0074	EDEADLK	A resource deadlock is avoided
117	0075	EEXIST	The file or socket exists
118	0076	EFAULT	The address is incorrect
119	0077	EFBIG	The file is too large
120	0078	EINTR	A function call is interrupted
121	0079	EINVAL	The parameter is incorrect
122	007A	EIO	An I/O error occurred
123	007B	EISDIR	The file specified is a directory
124	007C	EMFILE	Too many files are open for this process
125	007D	EMLINK	Too many links occurred
126	007E	ENAMETOOLONG	The filename is too long
127	007F	ENFILE	Too many files are open in the system
128	0080	ENODEV	No such device exists
129	0081	ENOENT	No such file or directory exists
130	0082	ENOEXEC	The exec call contained a format error
131	0083	ENOLCK	No locks are available
132	0084	ENOMEM	Not enough space is available
133	0085	ENOSPC	No space is left on the device
134	0086	ENOSYS	The function is not implemented

© Copyright IBM Corp. 1993, 2022

Table 5. OpenExtensions Return Codes by Numeric Value (continued)

Dec Value	Hex Value	Return Code	Description
135	0087	ENOTDIR	Not a directory
136	0088	ENOTEMPTY	The directory is not empty
137	0089	ENOTTY	The I/O control operator is inappropriate
138	008A	ENXIO	No such device or address exists
139	008B	EPERM	The operation is not permitted
140	008C	EPIPE	The pipe is broken
141	008D	EROFS	The specified file system is read only
142	008E	ESPIPE	The seek is incorrect
143	008F	ESRCH	No such process or thread exists
144	0090	EXDEV	A link to a file on another file system was attempted
145	0091	E2BIG	The parameter list is too long
146	0092	ELOOP	A loop is encountered in symbolic links
147	0093	EILSEQ	The byte sequence is illegal
151	0097	ECMSSTORAGE	Storage management error
156	009C	ECMSINITIAL	Process Initialization error
157	009D	ECMSERR	A CMS environmental or internal error has occurred
159	009F	ECMSPFSFILE	The physical file system encountered a permanent file error
162	00A2	ECMSPFSPERM	The physical file system encountered a system error
227	00E3	EBUFLEN	Buffer not long enough for path name
228	00E4	EEXTLINK	The target of the operation is an external link
229	00E5	ENODD	No pathdef for the ddname in effect
230	00E6	ECMSESMERR	CMS ESM error
231	00E7	ECPERR	CP DIAGNOSE error
1002	03EA	EIBMSOCKOUTOFRANGE	The socket number assigned by the client interface code is out of range.
1003	03EB	EIBMSOCKINUSE	The socket number assigned by the client interface code is already in use.
1005	03ED	EOFFLOADboxERROR	Offload box error.
1008	03F0	EIBMCONFLICT	A conflicting call is already outstanding on the socket.
1009	03F1	EIBMCANCELLED	The request has been cancelled by a SOCKcallCANCEL request.
1100	044C	ENOTBLK	A block device is required.
1101	044D	ETXTBSY	The text file is busy.

Table 5. OpenExtensions Return Codes by Numeric Value (continued)

Dec Value	Hex Value	Return Code	Description
1102	044E	EWOULDBLOCK	The descriptor is marked nonblocking, and the requested function cannot complete immediately.
1103	044F	EINPROGRESS	The operation is now in progress.
1104	0450	EALREADY	An operation is already in progress.
1105	0451	ENOTSOCK	A socket operation has been requested on a nonsocket.
1106	0452	EDESTADDRREQ	A destination address is required.
1107	0453	EMSGSIZE	The message is too large to be sent all at once, as required.
1108	0454	EPROTOTYPE	The socket type is incorrect.
1109	0455	ENOPROTOOPT	The protocol or socket option is not available.
1110	0456	EPROTONOSUPPORT	The protocol is not supported.
1111	0457	ESOCKTNOSUPPORT	The socket type is not supported.
1112	0458	EOPNOTSUPP	The referenced socket is not a type that supports the requested function.
1113	0459	EPFNOSUPPORT	The protocol family is not supported.
1114	045A	EAFNOSUPPORT	The address family is not supported.
1115	045B	EADDRINUSE	The address is already in use.
1116	045C	EADDRNOTAVAIL	Cannot assign the requested address.
1117	045D	ENETDOWN	The network is down.
1118	045E	ENETUNREACH	The network is unreachable.
1119	045F	ENETRESET	The network dropped the connection on reset.
1120	0460	ECONNABORTED	The software caused the connection to abort.
1121	0461	ECONNRESET	The connection was reset by the peer.
1122	0462	ENOBUFS	Insufficient buffer space is available.
1123	0463	EISCONN	The socket is already connected.
1124	0464	ENOTCONN	The socket is not connected.
1125	0465	ESHUTDOWN	Cannot send after a socket shutdown.
1126	0466	ETOOMANYREFS	Tthere are too many references — cannot splice.
1127	0467	ETIMEDOUT	The connection timed out.
1128	0468	ECONNREFUSED	The connection attempt was rejected.
1129	0469	EHOSTDOWN	The host is down.
1130	046A	EHOSTUNREACH	There is no route to the host.
1131	046B	EPROCLIM	There are too many processes.
1132	046C	EUSERS	There are too many users.

Table 5. OpenExtensions Return Codes by Numeric Value (continued)

Dec Value	Hex Value	Return Code	Description
1133	046D	EDQUOT	The disk quota has been exceeded.
1134	046E	ESTALE	The NFS file handle is stale.
1135	046F	EREMOTE	There are too many remote levels in the path.
1136	0470	ENOSTR	The device is not a stream.
1137	0471	ETIME	The timer has expired.
1138	0472	ENOSR	There are no streams resources.
1139	0473	ENOMSG	No message of the desired type
1140	0474	EBADMSG	Trying to read an unreadable message.
1141	0475	EIDRM	The identifier has been removed.
1142	0476	ENONET	The machine is not on the network.
1143	0477	ERREMOTE	The object is remote.
1144	0478	ENOLINK	The link has been severed.
1145	0479	EADV	Advertise error.
1146	047A	ESRMNT	srmount error.
1147	047B	ECOMM	Communication error on send.
1148	047C	EPROTO	Protocol error.
1149	047D	EMULTIHOP	Protocol error.
1150	047E	EDOTDOT	Cross mount point.
1151	047F	EREMCHG	Remote address change.
1152	0480	ECANCELED	The asynchronous I/O request has been canceled.
1160	0488	ENOREUSE	Socket descriptor reuse is not supported.
28672	7000	EBindModError	Error code issued by the DMSBX2WR CMS Binder routine that is invoked by DMSBX1WR, the intercept routine to BPX1WRT. This routine will write either a standard or extended format CMS module file or invoke the real BPX1WRT routine to write a BFS program object.
28928	7100	EBindNXError	Error code issued by the DMSBX2WX CMS Binder routine that is invoked by DMSBX2WR to check if a non-executable linear program object can replace an existing module file.

OpenExtensions Return Codes Listed by Symbolic Name

Table 6. OpenExtensions Return Codes by Symbolic Name

Return Code	Decimal	Hex
EACCES	111	006F
EADDRINUSE	1115	045B

Table 6. OpenExtensions Return Codes by Symbolic Name (continued)

Return Code	Decimal	Hex
EADDRNOTAVAIL	1116	045C
EADV	1145	0479
EAFNOSUPPORT	1114	045A
EAGAIN	112	0070
EALREADY	1104	0450
EBADF	113	0071
EBADMSG	1140	0474
EBindModError	28672	7000
EBindNXError	28928	7100
EBUFLEN	227	00E3
EBUSY	114	0072
ECANCELED	1152	0480
ECHILD	115	0073
ECMSSTORAGE	151	0097
ECMSERR	157	009D
ECMSESMERR	230	00E6
ECMSINITIAL	156	009C
ECMSPFSFILE	159	009F
ECMSPFSPERM	162	00A2
ECOMM	1147	047B
ECONNABORTED	1120	0460
ECONNREFUSED	1128	0468
ECONNRESET	1121	0461
ECPERR	231	00E7
EDEADLK	116	0074
EDESTADDRREQ	1106	0452
EDOM	1	0001
EDOTDOT	1150	047E
EDQUOT	1133	046D
EEXIST	117	0075
EEXTLINK	228	00E4
EFAULT	118	0076
EFBIG	119	0077
EHOSTDOWN	1129	0469

Table 6. OpenExtensions Return Codes by Symbolic Name (continued)

Return Code	Decimal	Hex
EHOSTUNREACH	1130	046A
EIBMCANCELLED	1009	03F1
EIBMCONFLICT	1008	03F0
EIBMSOCKINUSE	1003	03EB
EIBMSOCKOUTOFRANGE	1002	03EA
EIDRM	1141	0475
EILSEQ	147	0093
EINPROGRESS	1103	044F
EINTR	120	0078
EINVAL	121	0079
EIO	122	007A
EISCONN	1123	0463
EISDIR	123	007B
ELOOP	146	0092
EMFILE	124	007C
EMLINK	125	007D
EMSGSIZE	1107	0453
EMULTIHOP	1149	047D
ENAMETOOLONG	126	007E
ENETDOWN	1117	045D
ENETRESET	1119	045F
ENETUNREACH	1118	045E
ENFILE	127	007F
ENOBUFS	1122	0462
ENODD	229	00E5
ENODEV	128	0080
ENOENT	129	0081
ENOEXEC	130	0082
ENOLCK	131	0083
ENOLINK	1144	0478
ENOMEM	132	0084
ENOMSG	1139	0473
ENONET	1142	0476
ENOPROTOOPT	1109	0455

Table 6. OpenExtensions Return Codes by Symbolic Name (continued)

Return Code	Decimal	Hex
ENOREUSE	1160	0488
ENOSPC	133	0085
ENOSR	1138	0472
ENOSTR	1136	0470
ENOSYS	134	0086
ENOTBLK	1100	044C
ENOTCONN	1124	0464
ENOTDIR	135	0087
ENOTEMPTY	136	0088
ENOTSOCK	1105	0451
ENOTTY	137	0089
ENXIO	138	008A
EOFFLOADboxERROR	1005	03ED
EOPNOTSUPP	1112	0458
EPERM	139	008B
EPFNOSUPPORT	1113	0459
EPIPE	140	008C
EPROCLIM	1131	0459
EPROTO	1148	047C
EPROTONOSUPPORT	1110	0456
EPROTOTYPE	1108	0454
ERANGE	2	0002
EREMCHG	1151	047F
EREMOTE	1135	046F
EROFS	141	008D
ERREMOTE	1143	0477
ESHUTDOWN	1125	0465
ESOCKTNOSUPPORT	1111	0457
ESPIPE	142	008E
ESRCH	143	008F
ESRMNT	1146	047A
ESTALE	1134	046E
ETIME	1137	0471
ETIMEDOUT	1127	0467

Table 6. OpenExtensions Return Codes by Symbolic Name (continued)

Return Code	Decimal	Hex
ETOOMANYREFS	1126	0466
ETXTBSY	1101	044D
EUSERS	1132	046C
EWOULDBLOCK	1102	044E
EXDEV	144	0090
E2BIG	145	0091

Appendix B. Reason Codes

This appendix describes the reason codes returned by OpenExtensions callable services. Only the character value is intended as an interface for programmers.

This appendix contains three sections:

- "OpenExtensions Reason Codes Listed by Numeric Value" on page 495. This list describes the action required to correct each error.
- "Special CMS File Pool Server and BFS Client Reason Codes" on page 532.
- "OpenExtensions Reason Codes Listed by Symbolic Name" on page 534. In this list, reason code names are cross-referenced to reason code values.

OpenExtensions Reason Codes Listed by Numeric Value

The reason code is made up of 4 bytes in the format *ccccrrrr*, where:

cccc
is a halfword reason code qualifier

rrrr
is the halfword reason code

The two high-order bytes of the reason codes returned by OpenExtensions services contain a value that is used to qualify the contents of the two low-order bytes. If the contents of the two high-order bytes is within the range 0000 to X'21FF', the error represented by the reason code is defined by OpenExtensions. If the contents of the two high-order bytes is outside the range, the error represented by the reason code is not an OpenExtensions reason code.

Use Table 7 on page 495 to determine where you can find information on the reason codes returned by callable services.

Table 7. Location of Return Information

Return Code	Reason Code Qualifier	Reason Code Returned By
A3	0000–21FF	External security manager (ESM). See the specific ESM service for the meaning of these reason codes.
A2	5B00–5BFF	File pool server and BFS client. See "Special CMS File Pool Server and BFS Client Reason Codes" on page 532 for descriptions of these reason codes.
7000, 7100	8300	z/VM: Program Management Binder for CMS utilizes POSIX callable services to process program object data. When errors are detected an error message is generated, and then the following z/OS® MVS™ Program Management Binder message (containing the OpenExtensions reason code, *rsn*) is issued: **IEW2796S** DF16 FILE ASSOCIATED WITH DDNAME */fd* CANNOT BE WRITTEN. HFS WRITE ISSUED RETURN CODE *rc* AND REASON CODE *rsn*. The following table contains descriptions of these reason codes.

© Copyright IBM Corp. 1993, 2022

Table 7. Location of Return Information (continued)

Return Code	Reason Code Qualifier	Reason Code Returned By
All (except A2, A3)	0000–21FF	OpenExtensions. The following table contains descriptions of these reason codes.

Table 8. OpenExtensions Reason Codes by Numeric Value

Dec Value	Hex Value	Description
0	0000	**JROK:** The return code value describes the error. *Action:* Refer to the return code for information on the error.
37	0025	**JRUnexpectedErr:** An unexpected error occurred. *Action:* See your IBM service representative.
40	0028	**JRMaxProc:** The maximum number of processes was exceeded. *Action:* Retry after some processes have ended.
46	002E	**JRFilesysNotThere:** The file system named does not exist. *Action:* The file system specified on the service could not be found.
48	0030	**JRNegativeValueInvalid:** A negative value cannot be supplied for one of the parameters. *Action:* Enter the call again after changing the offending parameter to a valid value.
50	0032	**JrUnlMountRO:** The unlink call was on a read-only file system. *Action:* For the file to be unlinked, the file system must be mounted in read/write mode.
51	0033	**JRRFileWrOnly:** A call tried to read a file opened as write-only. *Action:* Reopen the file for read or read/write access.
52	0034	**JRWFileRdOnly:** A call tried to write to a file opened as read-only. *Action:* Reopen the file for write or read/write access.
54	0036	**JRNegFileDes:** A negative file descriptor was requested. *Action:* Reissue the request with a nonnegative file descriptor.
55	0037	**JRFileDesNotInUse:** The requested file descriptor is not in use. *Action:* Reissue the request with an open file descriptor.
56	0038	**JRMkDirExist:** The requested file directory already exists. *Action:* A directory by this name exists. The MKDIR request cannot be processed. Correct the name and retry the operation.
57	0039	**JRPathTooLong:** The path name is too long. *Action:* The path name was found to be larger than PATH_MAX (1023). Either the name specified was too long, or the name generated as a result of using symbolic links was too long. Correct the name and retry the operation.
58	003A	**JRNullInPath:** The path name contains a null. *Action:* Check the path name specified to find and remove the embedded null. If the request was for a symbolic link, there must be no nulls within the contents of the symbolic link.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
59	003B	**JRNotSysRoot:** A relative path name is allowed only for processes. *Action:* See your IBM service representative.
60	003C	**JRCompNotDir:** A node in the path name is not a directory. *Action:* One of the components of the path name was found to not be a directory. All but the final component of the name must be directories. Correct the path name and retry the operation.
61	003D	**JRDirNotFound:** A directory in the path name was not found. *Action:* One of the directories specified was not found. Verify that the name specified is spelled correctly.
62	003E	**JRCompNameTooLong:** A component in the path name was too long. *Action:* One of the components of the path name was found to be larger than NAME_MAX (255). Correct the path name and retry the operation.
63	003F	**JRInvOpenFlags:** The open call detected incorrect open flags. *Action:* The OPEN request cannot be processed. Correct the open flags and retry the operation.
65	0041	**JRTrNotRegFile:** The ftruncate call is valid only on a regular file. *Action:* To be able to truncate a file, you must specify the File_descriptor for a file, not for a directory or a FIFO.
66	0042	**JRClNeedClose:** The closedir call was for a file that was opened with the open call. *Action:* Retry the request, using CLOSE.
67	0043	**JRPfsDead:** The file system owning the file is no longer active. *Action:* See your IBM service representative.
68	0044	**JRMkDir:** The mkdir service is not supported by this file system. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
69	0045	**JRClose:** Vnode operation CLOSE is not supported by this file system. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
70	0046	**JRRdwr:** Vnode operation RDWR is not supported by this file system. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
71	0047	**JRLookup:** Lookup is not supported by this file system. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
73	0049	**JRVnodGet:** A cell pool get for a vnode failed. *Action:* See your IBM service representative.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
75	004B	**JROpen:** The open service is not supported by this file system. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
76	004C	**JRCreate:** The create service is not supported by this file system. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
77	004D	**JRNoPath:** The path length is not greater than 0. *Action:* There must be a positive length passed for the path name length.
79	004F	**JRChdNotDir:** The chdir service is valid only for directory files. *Action:* Reissue the chdir service specifying, the name of a directory file.
80	0050	**JRChdNoEnt:** The chdir service was invoked with the name of a nonexisting file. *Action:* Reissue the chdir service, specifying the name of an existing directory file.
85	0055	**JRMkDirROnly:** The directory cannot be created in a read-only file system. *Action:* The file system was mounted read-only. The mkdir service request cannot be processed.
86	0056	**JRLnkDir:** Hard links cannot be made to directory files. *Action:* Use the symlink service to create a symbolic link to the desired directory.
87	0057	**JRLskOnPipe:** The lseek service cannot be performed on a pipe. *Action:* The lseek service must be performed on either a regular file or a directory.
88	0058	**JRLskOffsetIsInvalid:** The offset given for lseek service is incorrect. *Action:* The final cursor value on an lseek call cannot be a negative number. If the Reference_point specified "Set" the offset must be nonnegative. If the Reference_point specified "Current", then the sum of the input offset and the current cursor value must be nonnegative. If the Reference_point specified "End", then the sum of the input offset and the cursor value of the end of the file must be nonnegative.
89	0059	**JRLskWhenceIsInvalid:** The whence given for the lseek service is incorrect. *Action:* The lseek operation can specify a Reference_point of either "Set", "Current", or "End".
90	005A	**JRFSNotStart:** The specified file system type is not supported. *Action:* The file_system_type specified on a mount request must be VMBFS.
91	005B	**JRIsMounted:** The file system is already mounted. *Action:* If the file system must be mounted on the specified mountpoint, first unmount it, and then reissue the request.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
92	005C	**JRMountPt:** A problem was found with the mount point specified. *Action:* The problem found depends on the return code received with this reason code. • If the return code is ENOENT, the path name specified could not be found. • If the return code is ENOTDIR, the path name did not specify a directory. • If the return code is EINVAL, the path name specified refers to the root of an already mounted file system.
93	005D	**JRUnlNoEnt:** The file to unlink does not exist. *Action:* The file has either already been unlinked, or it never existed. Verify that the path name was correct.
94	005E	**JRUnlDir:** The unlink service was requested on a directory file. *Action:* To remove a directory, use the rmdir service.
95	005F	**JROutOfOfteCells:** The system was unable to obtain a cell from the OFTE cell pool. *Action:* See your IBM service representative.
96	0060	**JRReadOnlyFileSetWriteReq:** An open request for write was entered for a file system that was mounted read-only. *Action:* The open service request cannot be processed. Mount the file system for read-write access and reissue the open request.
97	0061	**JRReadOnlyFileSetCreatReq:** A file cannot be created in a read-only file system. *Action:* The file system was mounted read-only. The open create service request cannot be processed.
98	0062	**JRNoFileNoCreatFlag:** A service tried to open a nonexistent file without O_CREAT. *Action:* The open service request cannot be processed. Correct the name or the open flags, and retry the operation.
99	0063	**JRFileExistsExclFlagSet:** The file exists, but O_EXCL is specified on the open call. *Action:* The open service request cannot be processed. Correct the name or the open flags, and retry the operation.
100	0064	**JRDirWriteRequest:** The service tried to open a directory for write access. *Action:* The open service request cannot be processed. Correct the name or the open flags, and retry the operation.
101	0065	**JROpenFlagConflict:** The call tried to open a file with O_RDONLY and O_TRUNC specified. *Action:* The open service request cannot be processed. Correct the open flags and retry the operation.
103	0067	**JRParmTooLong:** On the mount, a parameter field longer than 1024 was specified. *Action:* Specify a parameter length not be longer than 1024 bytes.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
104	0068	**JRRemove:** Vn_Remove is not supported by the physical file system. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
105	0069	**JRBothMode:** On the mount service, both read and read/write were specified. *Action:* The Mount_mode on a mount service cannot specify both read-write and read-only.
106	006A	**JRNeitherMode:** On the mount service, neither read nor read/write were specified. *Action:* The Mount_mode on a mount service must specify either read-write and read-only.
107	006B	**JRBuffTooSmall:** The buffer for return information is too small. *Action:* The length of the buffer specified on the service was not large enough to contain the data to be returned.
108	006C	**JRFileNotThere:** The requested file does not exist. *Action:* The service cannot be performed unless the named file exists.
109	006D	**JRReadDir:** The readdir service vnode operation is not supported. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
110	006E	**JRGetAttr:** GetAttr is not supported by this file system. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
112	0070	**JRRddFileNotDir:** The readdir service request was on a file that was not opened as a directory. *Action:* Use the opendir service to open the directory.
113	0071	**JRTargetNotDir:** The opendir service did not specify a directory. *Action:* The opendir service request cannot be processed. Correct the name and retry the operation.
114	0072	**JROpenDirNotFound:** The directory specified on the opendir service did not exist. *Action:* The opendir service request cannot be processed. Correct the name and retry the operation.
117	0075	**JRSpFileExists:** The file specified on the mknod service already existed. *Action:* A file by this name exists. The mknod service request cannot be processed. Correct the name and retry the operation.
118	0076	**JRReadOnlyFileSetMknodReq:** A special file cannot be created on a read-only file system. *Action:* Specify another file system or unmount and remount the current file system.
119	0077	**JRRmDir:** The rmdir service vnode operation is not supported. *Action:* Verify that the operation was performed on a physical file system that supports the operation.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
120	0078	**JRPathNotDir:** The path name does not specify a directory. *Action:* The service requested requires a directory, but the path name passed is not for a directory.
121	0079	**JRReadOnlyFS:** This operation does not work on a read-only file system. *Action:* The service was requested for a file system that was mounted read-only. The service requires that the file system be mounted read/write.
123	007B	**JRDiffFileSets:** The rename service is not supported across file systems. *Action:* The rename service cannot be performed across file systems. Rename the file, specifying a new name within the same file system.
124	007C	**JRNewNotDir:** The new name specified on the rename service is not a directory. *Action:* If a directory is to be renamed to an existing file name, that file name must refer to a directory file.
125	007D	**JRNewIsDir:** The new name specified on the rename service is a directory. *Action:* If a nondirectory is to be renamed to an existing file name, that file name must not refer to a directory file.
126	007E	**JROldNoExist:** The old name specified on the rename service does not exist. *Action:* The file to be renamed does not exist. Reissue the request with an existing file name.
127	007F	**JRIsFSRoot:** The name specified is in use as a file system root. *Action:* The function cannot be performed on the root of the file system.
128	0080	**JRRename:** The rename service vnode operation is not supported. *Action:* See your IBM service representative.
130	0082	**JRDotOrDotDot:** The requested function cannot be performed against . or ... *Action:* Neither `.` nor `..` can be specified for this operation.
132	0084	**JRInternalError:** An internal error was detected. *Action:* See your IBM service representative.
134	0086	**JRBadEntryCode:** An incorrect entry code was specified on this request. *Action:* A command code or entry code specified on the request is not correct. Reissue the command using a valid command code.
136	0088	**JRFdAllocErr:** An error occurred while trying to allocate a filedes page. *Action:* Close any file descriptors that are no longer needed.
138	008A	**JRBytes2RWZero:** The number of bytes requested to read or write is negative. *Action:* Specify a positive number for the number of bytes to be read or written.
139	008B	**JRRwdFileNotDir:** The rewinddir service was on a file that is not a directory. *Action:* The rewinddir service requires that the file descriptor passed on input refer to a directory.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
140	008C	**JRRootNode:** The requested operation cannot be done on a root. *Action:* The function was requested for a file system root, but it cannot be done on a root.
141	008D	**JRInvalidSignal:** A signal number specified is incorrect. *Action:* Reissue the request with a valid signal number.
142	008E	**JRInvalidSigAct:** The action is incorrect for the specified signal. *Action:* Reissue the request with a valid signal action.
143	008F	**JRInvalidSigHow:** The how operand specified is incorrect. *Action:* Reissue the request with a valid how operand.
144	0090	**JRNotForDir:** The system cannot perform the requested function on a directory. *Action:* The file descriptor specified refers to a directory file, and the request is not valid for such a file descriptor. Reissue the request specifying a nondirectory file descriptor.
145	0091	**JROldPartOfNew:** The old name specified on the rename service is part of the new name. *Action:* Reissue the rename request, specifying a new name that does not contain the old name.
156	009C	**JRTrOpenedRO:** The ftruncate service was for a file opened in read-only mode. *Action:* To be able to truncate a file, you must open it for write.
157	009D	**JRTrMountedRO:** The ftruncate service was for a file on a file system mounted in read-only mode. *Action:* For you to be able to truncate a file, it must not be on a file system that has been mounted in read-only mode.
158	009F	**JRTrNegOffset:** A negative offset was given to the ftruncate service. *Action:* To truncate a file, specify a nonnegative File_length.
160	00A0	**JROutOfLocks:** The file system has run out of locks. *Action:* When a file system lock was requested, there were no more left in the system. Try again later.
161	00A1	**JRMount:** The mount service VFS operation is not supported. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
162	00A2	**JRUMount:** The unmount service VFS operation is not supported. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
164	00A4	**JRRoot:** The Root VFS operation is not supported. *Action:* Verify that the operation was performed on a physical file system that supports the operation.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
168	00A8	**JRInvalidVnode:** The vnode returned is not valid. *Action:* See your IBM service representative.
169	00A9	**JRInvalidParms:** An incorrect combination of parameters was specified. *Action:* The actual problem depends on the service. • For a mount service, Mount_mode must not specify any values unrelated to a mount service. • For an unmount service, Flags must not specify any values unrelated to an unmount service, and must not specify mutually exclusive requests. • For all others, correct the specified parameters and reissue the request.
175	00AF	**JRLockErr:** The file system had a lock error. *Action:* See your IBM service representative.
176	00B0	**JRUserNotPrivileged:** The requester of the service is not privileged. *Action:* The requested service required a privileged user. Check the documentation for the service to understand what privilege is required.
177	00B1	**JRUnexpectedError:** An unexpected return value was received. *Action:* See your IBM service representative.
180	00B4	**JRQuiesced:** There was a previous quiesce request. *Action:* The file system required for the current function has been quiesced. After the file system has been unquiesced, retry this service.
182	00B6	**JRPfsSuspend:** The physical file system needs to be restarted. *Action:* Contact your IBM service representative.
184	00B8	**JRNoStorage:** Error obtaining free storage. *Action:* You must either free some virtual storage or increase the size of your virtual machine. To increase the size of your virtual machine, use the DEFINE command; then reIPL CMS and enter the original command again.
256	0100	**JRTrunc:** Vnode operation trunc is not supported by this file system. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
257	0101	**JRFsync:** Vnode operation fsync is not supported by this file system. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
258	0102	**JRSetAttr:** Vnode operation setattr is not supported by this file system. *Action:* Verify that the operation was performed on a physical file system that supports the operation.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
259	0103	**JRSymFileAlreadyExists:** The file requested for creation as a symbolic link already exists. *Action:* The link name specified on a symlink service request is an existing file name. Reissue the request specifying a link name that does not already exist.
260	0104	**JRSymlink:** The symbolic link vnode operation is not supported. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
261	0105	**JRFileNotSymLink:** The file requested for readlink service is not a symbolic link. *Action:* Reissue the readlink service request specifying the name of a file other than a symbolic link.
262	0106	**JRReadlink:** The readlink vnode operation is not supported. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
263	0107	**JRMknodInvalidType:** The mknod service invoked with incorrect file type parameter. *Action:* The type specified in the mknod service is not supported. The service cannot be processed. The mknod service accepts only FT_CHARSPEC and FT_FIFO. See BPXYFTYP. Correct the type parameter and retry the operation.
264	0108	**JREndingSlashMknod:** The path name ended with slash on the mknod service. *Action:* The path name specified for a mknod service request ended with a slash. The service request cannot be processed. Correct the name and retry the operation.
265	0109	**JREndingSlashOCreat:** The path name ended with slash on the open o_creat service. *Action:* The open service request cannot be processed. Correct the name and retry the operation.
266	010A	**JRLnkNoEnt:** The service tried to link to nonexistent file. *Action:* Use the open service to create the file, or reissue the request specifying an existing file name.
267	010B	**JRLnkNewPathExists:** The service tried to add a link whose name already exists. *Action:* Reissue the request, specifying a new path name that does not already exist.
268	010C	**JRLnkAcrossFilesets:** The service tried to link across file systems. *Action:* Reissue the request, specifying a new path name that is within the same file system as the existing path name.
269	010D	**JRLnkROFileset:** The service tried to add a directory entry on a read-only file system. *Action:* For you to create a link to the existing path name, the file system must be mounted in read-write mode.
270	010E	**JRLink:** Vn_Link is not supported by this physical file system. *Action:* Verify that the operation was performed on a physical file system that supports the operation.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
271	010F	**JRExecNmLenZero:** The length of the executable name passed was zero. *Action:* The parameter specifying the length of the program name to be run contained zero. Correct the program name length and resubmit the job.
274	0112	**JRFsFailStorage:** Spawn failed, due to unavailable file system storage. *Action:* See your IBM service representative.
276	0114	**JRNotPermitted:** You are not permitted to signal to the specified process ID (PID). *Action:* Reissue the request specifying a PID that you are authorized to send a signal to, or reissue the request from a superuser ID.
277	0115	**JRBuffLenInvalid:** The length of the buffer is less than or equal to zero. *Action:* The buffer length specified for this request was either a negative number, or was zero. Retry the request specifying a valid buffer length parameter.
281	0119	**JRNotSupportedForFileType:** The requested service is not supported for this file type. *Action:* Reissue the request, specifying a file of the correct type for the request.
282	011A	**JRInvalidSymLinkLen:** The contents specified for the symbolic link has an incorrect length. *Action:* Reissue the symlink request specifying a path length greater than or equal to zero, but less than 1023.
283	011B	**JRInvalidSymLinkComp:** The contents specified for symbolic link has an incorrect component. *Action:* The contents of a symbolic link must consist of components whose length cannot exceed 255 characters.
284	011C	**JRFileNotOpen:** The file is not opened. *Action:* Reissue the request specifying an open file descriptor.
285	011D	**JRTooManySymlinks:** Too many symbolic links were encountered in the path name. *Action:* While attempting to resolve the input path name, more than POSIX_SYMLOOP (8) symbolic links were found.
287	011F	**JRExecNotRegFile:** The file name specified on the exec is not a regular file. *Action:* The exec service detected that the file name specified by the path name argument is not a regular type file. Correct the path name argument and resubmit the job.
290	0122	**JRInactive:** The vnode operation inactive is not supported by the file system. *Action:* See your IBM service representative.
291	0123	**JRInvalidMajorNumber:** Character special file system detected an incorrect device major number. *Action:* This character special file is not supported by any device drivers installed on this system. The request cannot be processed. Correct the path name and retry the request.
293	0125	**JRRdandWRTforPipe:** The open call on a pipe was for read/write. *Action:* The request cannot be processed. Correct the open flags and retry the request.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
296	0128	**JROpenforWriteNoReaders:** Open for write was done before any open for read. *Action:* Open for write was requested while file flags indicated O_NONBLOCK and before any open for read. The request cannot be processed. An open for read request must precede an open for write request.
297	0129	**JRNoReaders:** The service tried to write before any open for reads. *Action:* An open for read must be performed.
301	012D	**JRSyscallAbend:** An abend occurred in a system call. *Action:* See your IBM service representative.
302	012E	**JRBadAddress:** An incorrect address was encountered when the system tried to move data. *Action:* An error occurred while the system was accessing the user data. Check for incorrect input parameters passed to the system call.
304	0130	**JRSigDuringWait:** A signal occurred during a wait. *Action:* While the service was waiting for a to be performed, a signal was received to interrupt it.
307	0133	**JRRdnorWRTforPipe:** The open service on a pipe was for neither read nor write. *Action:* The request cannot be processed. Correct the open flags and retry the service.
309	0135	**JRNoData:** There is no data in this pipe. *Action:* Try this service again later.
310	0136	**JRUserNotAuthorized:** The user is not authorized for the requested file descriptor. *Action:* When the specified file descriptor was opened, the user was executing in an authorized state. However, the user is now no longer authorized to use this file descriptor. Reissue the request, specifying a file descriptor to which the user has authority.
312	0138	**JRFileIsBlocked:** The file is blocked. *Action:* The request cannot be processed. Try again later.
313	0139	**JRIoctl:** The ioctl service is not supported by this file system. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
314	013A	**JRInvalidPid:** The process ID (PID) was not found, and the signal was not sent. *Action:* The target PID was either ended, or it never existed. Retry the function with an existing PID.
319	013F	**JRInvTermStat:** An incorrect process termination status was passed to BPX1MPC. *Action:* Structure BPXYWAST describes the valid terminating status.
324	0144	**JRSignalsNotBlocked:** The service is not completed, and signals are not blocked. *Action:* This service can be run only if all signals are blocked.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
325	0145	**JRFdTooBig:** The requested file descriptor exceeds the DFLT_OPEN_MAX limit. *Action:* Reissue the request, specifying a file descriptor that does not exceed the DFLT_OPEN_MAX limit.
326	0146	**JROpenMax:** The maximum number of open files for this process was reached. *Action:* Close any file descriptors that are no longer needed.
329	0149	**JRIOBufLengthInvalid:** The input argument to the buffer length was incorrect. *Action:* The argument length specified for this request was either a negative number, or was greater than 1024. Retry the request specifying a valid argument length parameter.
330	014A	**JRInvalidAmode:** An incorrect access mode was specified on the access service. *Action:* The access mode specified on the access service either had none of the valid flags turned on, or it had unsupported bits turned on. Reissue the request specifying a valid access mode.
331	014B	**JRAccess:** The access vnode operation is not supported. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
336	0150	**JRBadAuditOption:** An incorrect option code was specified for the chaudit service. *Action:* Reissue the request specifying a valid audit option code.
337	0151	**JRExecFileTooBig:** The size of the specified file exceeds the available virtual machine storage. *Action:* The exec service has detected that the size of the executable to be run exceeds the available virtual machine storage.
342	0156	**JRSignalReceived:** The call was interrupted by a signal. *Action:* A signal was received while this callable service was blocked. Retry the service, if appropriate.
359	0167	**JRFuncNotSupported:** The function is not supported by device driver. *Action:* See your IBM service representative.
391	0187	**JRChowntoPipe:** The fchown service was issued against a pipe. *Action:* This request cannot be performed against a pipe. Select a file descriptor that refers to a nonpipe file and reissue the request.
392	0188	**JRChaudtoPipe:** The fchaudit service was issued against a pipe. *Action:* This request cannot be performed against a pipe. Select a file descriptor that refers to a nonpipe file and reissue the request.
394	018A	**JRWrongSsave:** The caller's SVC level was incorrect. *Action:* A function was requested that requires the user to be running at the SVC level at which the thread was created or at which the cmssigsetup (BPX1MSS) service was issued. The condition is probably the result of issuing a service sensitive to SVC level after performing an operation such as CMSCALL or LINK that creates a new SVC level.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
399	018F	**JRQuiescing:** The call did not complete. The file system is unmounting. *Action:* The requested function cannot be performed while an unmount is in progress for a file system. Retry when the file system is mounted again.
408	0198	**JRInvIoctlCmd:** The w_ioctl (BPX1IOC) service was invoked with an incorrect command. *Action:* The command is not supported. Correct the program and rerun.
425	01A9	**JRNoCTTY:** There is no controlling terminal for this process. *Action:* The open request cannot be processed.
430	01AE	**JRBrlmNotActive:** The byte range lock manager is not active. *Action:* The byte range lock manager is trying to recycle. Reissue the request after the recycle has completed.
431	01AF	**JRBrlmFileLockRecycling:** File lock is being recycled; do not use until the file is closed by all users. *Action:* The requested file can no longer be used for byte range locking. The file cannot be locked until it has been recycled. To recycle the file, close all file descriptors open for this file. After all users have closed this file, it will be recycled. It may be some time before all open file descriptors for this file have been closed.
432	01B0	**JRBrlmBadFileType:** Byte range locking can be performed only on regular files. *Action:* Reissue the request specifying the file descriptor for a regular file.
433	01B1	**JRBrlmNoReadAccess:** Shared byte range locks are only for files open for read. *Action:* To set a read lock on a file, it must be opened with read access. Reissue the request specifying a file descriptor that has read access to the file.
434	01B2	**JRBrlmNoWriteAccess:** Exclusive byte range locks are only for files open for write. *Action:* To set a write lock on a file, it must be opened with write access. Reissue the request specifying a file descriptor that has write access to the file.
435	01B3	**JRBrlmBadL_Type:** A byte range lock request specified an l_type that is not valid. *Action:* The value specified for l_type must be one of the following, found in BPXYBRLK: • F_RDLCK to set a read lock • F_WRLCK to set a write lock • F_UNLCK to unlock a range
436	01B4	**JRBrlmInvalidRange:** A byte range lock extends to before the start of the file. *Action:* The range specified by the l_start, l_whence, and l_len must not extend beyond the beginning of the file. Reissue the request specifying a valid range.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
437	01B5	**JRBrlmBadL_Whence:** A byte range lock request specified an l_whence that is not valid. *Action:* The value specified for l_whence must be one of the following, found in BPXYSEEK: • SEEK_SET • SEEK_CUR • SEEK_END
439	01B7	**JRBrlmRangeNotAvailable:** All or part of requested range is held by another user. *Action:* Issue a set lock with wait request to obtain the requested lock when all current users and waiters have freed it.
440	01B8	**JRBrlmDeadLockDetected:** Waiting on the specified range will cause a deadlock. *Action:* To avoid deadlock, release the locks being held before requesting a new range. All users should obtain locks in the same order to maintain a lock hierarchy and avoid deadlocking.
441	01B9	**JRBrlmSignalPosted:** While the process was waiting for a byte range lock, a signal was posted. *Action:* A signal was posted while the process was waiting for a lock. The lock is not obtained.
445	01BD	**JRBrlmBadL_Len:** A byte range lock request specified an incorrect l_len. *Action:* The l_len value cannot be less than zero. Reissue the request specifying an l_len that is greater than or equal to zero.
450	01C2	**JRBrlmAlreadyWaiting:** Request includes a range already being waited on. *Action:* The process is already waiting for a byte range lock that intersects with the requested range. Wait until the first request is honored before issuing another.
451	01C3	**JRBrlmPromotePending:** Another user is waiting to promote the requested range. *Action:* Another user has already requested promotion of the requested range. That promotion will not be granted until all other users unlock their shared locks on that range. Unlock the range in conflict and issue a set-lock-wait request for the exclusive lock desired.
453	01C5	**JRBrlmProcessBroken:** This process has been marked broken for byte locking. *Action:* The process may no longer issue byte range locking requests.
457	01C9	**JRBrlmUnlockWhileWait:** The unlock service is not valid while the process is waiting for a lock. *Action:* The process is presently waiting for a lock. No unlock requests will be accepted while the process is waiting.
458	01CA	**JRBrlmObjAndProcBroken:** The object and process are marked broken for byte locking. *Action:* The process can no longer issue byte-range-locking requests.
461	01CD	**JRFd2TooSmall:** The second file descriptor cannot be smaller than the first. *Action:* The specified request requires that the second file descriptor be greater than or equal to the first file descriptor.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
462	01CE	**JRPtCreateError:** An unexpected error occurred in the BPX1PTC service. *Action:* See your IBM service representative.
464	01D0	**JRPtExitError:** An unexpected error occurred in the BPXPTEXT service. *Action:* See your IBM service representative.
465	01D1	**JRPtCancelError:** An unexpected error occurred in the BPX1PTB service. *Action:* See your IBM service representative.
467	01D3	**JRPtatEye:** The pthread attribute area contains an incorrect eyecatcher. *Action:* The eyecatcher value must be BPXYPTAT. Reissue the BPX1PTC callable service with the corrected eyecatcher value.
470	01D6	**JRAllFilesNotClosed:** All requested files were not closed. *Action:* Some of the file descriptors within the specified range remain open. Use closedir to close any directory file descriptors. Any other file descriptors that remain open may have been opened while the process was executing in an authorized state, and the process may no longer be authorized to use them.
472	01D8	**JRThreadTerm:** The service rejected, and the requesting thread is in termination. *Action:* The BPX1PTX callable service should be issued to complete the termination of the thread and to obtain a new thread to process. All other OpenExtensions callable services are not supported while a thread is in this state.
474	01DA	**JRLightWeightThid:** The thread specified is a lightweight thread. *Action:* The thread specified by the caller is for a lightweight thread. Lightweight threads are not managed by OpenExtensions.
475	01DB	**JRAlreadyDetached:** The thread specified is already detached. *Action:* The thread specified by the caller is already detached. The requested service cannot be performed on a detached thread.
476	01DC	**JRThreadNotFound:** The thread specified was not found. *Action:* The thread specified by the caller is not a thread in the current process known by OpenExtensions.
478	01DE	**JRHeavyWeight:** The new thread was not started, and the exiting thread is a heavyweight thread. *Action:* The existing task is a heavyweight thread and cannot be reused using the PTGetNewThread option.
479	01DF	**JRGetFirst:** The first call did not specify PTGetNewThread. *Action:* The first call to this service from a newly created thread must specify the PTGetNewThread option.
480	01E0	**JRAlreadyJoined:** The thread specified was already joined by another thread. *Action:* The thread specified by the caller of the pthread_join service is currently joined by another thread. The target thread of a pthread_join can be joined by only one thread at a time.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
483	01E3	**JRJoinToSelf:** The thread attempted to join to itself. *Action:* The thread specified by the caller and the thread calling the pthread_join service are the same. A thread is not allowed to join to itself.
488	01E8	**JRAlreadyTerminated:** The calling thread has already ended. *Action:* The thread specified by the caller of the pthread_cancel service has already been canceled or exited and is in the process of being ended.
490	01EA	**JRBrokenBrlmRecycling:** The byte-range-lock manager is broken and is currently recycling. *Action:* The byte range lock manager is trying to recycle. Reissue the request when the recycle has completed.
491	01EB	**JRPtatSysOff:** The system offset value in the pthread attribute area is incorrect. *Action:* The system offset value must be set to the value provided with the BPXYPTAT mapping, PTATSYSOFFVAL. Reissue the BPX1PTC callable service with the corrected system offset value.
492	01EC	**JRPtatSysLen:** The system length value in the pthread attribute area is incorrect. *Action:* The system length value must be set to the value provided with the BPXYPTAT mapping, PTATSYSLENVAL. Reissue the BPX1PTC callable service with the corrected system length value.
493	01ED	**JRPtatLen:** The total length value in the pthread attribute area is incorrect. *Action:* The total length value must be set to the sum of PTAT#LENGTH and PTATUSERLENGTH. Use the BPXYPTAT mapping to correct this error. Reissue the BPX1PTC callable service with the corrected total length value.
495	01EF	**JRInvOption:** Incorrect option specified on call to BPX1PTX. *Action:* The option specified is either not a supported option or is a supported option that was specified in an unsupported environment. Examples of the latter error are: • The PTGETNEWTHREAD option is specified from a thread that was not created through the BPX1PTC callable service. The PTEXITTHREAD option is supported only from this type of thread. • The PTEXITTHREAD option is specified on the first invocation of BPX1PTX from the thread initialization routine. The first invocation of BPX1PTX must specify the PTGETNEWTHREAD option to obtain the first thread to process. Reissue the BPX1PTX callable service with the corrected option value.
498	01F2	**JRPtatSyncType:** The pthread attribute area contains an incorrect Sync Type value. *Action:* The pthread sync type attribute value must be set to PTATSYNCHRONOUS. Use the BPXYPTAT mapping for the definition of this value. Reissue the BPX1PTC callable service with the corrected pthread sync type attribute value.
499	01F3	**JRPtatDetachState:** The pthread attribute area contains an incorrect detach state value. *Action:* The pthread detach state attribute value must be set to PTATUNDETACHED or PTATDETACHED. Use the BPXYPTAT mapping for the definition of these values. Reissue the BPX1PTC callable service with the corrected pthread detach state attribute value.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
500	01F4	**JRNoSuchPid:** The process ID is incorrect. *Action:* Choose a process ID that is known to OpenExtensions.
501	01F5	**JRPidEQSessLeader:** The process ID is a session leader. *Action:* Choose a process ID that is not a session group leader.
502	01F6	**JRTooMany:** The event list specified for cond_post contained more than one event. *Action:* The event list specified for the BPX1CPO callable service contained more than one event. Reissue the BPX1CPO callable service with an event list that contains only one event.
503	01F7	**JRPidDifferentSession:** The process ID is in a session different from the caller. *Action:* Choose a process ID that is in the same session as the caller.
504	01F8	**JRSetpgidAfterSpawn:** The process ID specified on SETPGID is child process created using the spawn service. *Action:* Choose a process ID that does not belong to a process that has been started with spawn, or specify the process group on the spawn call.
506	01FA	**JRNotDescendant:** The process ID is not a descendant of the caller. *Action:* Choose a process ID that is a descendant of the caller (that is, not a child or child of a child).
507	01FB	**JRPgidDifferentSession:** Process group ID is in a session different from the caller, or does not exist. *Action:* Choose a process group ID that is in the same session as the caller.
508	01FC	**JRCallerIsPgLeader:** The caller is already a process group leader. *Action:* Choose a process ID that is not already a process group leader.
510	01FE	**JRRdlBuffLenInvalid:** The length of the buffer is less than zero. *Action:* The readlink service requires that the specified buffer length be greater than or equal to zero.
513	0201	**JRAlreadySigSetUp:** BPX1MSS found the process already set up for signals. *Action:* Only one task can be set up for signals at any one time. Issue the signal unset (BPX1MSD) service on the task that did the last setup and then reissue this service.
514	0202	**JRNotSigSetUp:** The service found the current task was not set up for signals. *Action:* Issue the signal setup service BPX1MSS and then reissue this service.
515	0203	**JREndingSlashSymlink:** The path name ended with slash on the symlink service. *Action:* The link name specified on a symlink request contained a trailing slash. Reissue the request omitting the trailing slash.
516	0204	**JRUndefEvents:** The specified event list contains undefined events. *Action:* Only specify events defined in BPXYCW for the BPX1CSE, BPX1CWA, or BPX1CTW callable services. For the BPX1CPO service, the only event allowed is CW_CONDVAR.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
517	0205	**JRNoEvents:** The specified event list is zero. *Action:* Specify one or more events defined in BPXYCW for the BPX1CSE, BPX1CWA, or BPX1CTW callable services. For the BPX1CPO service, specify the CW_CONDVAR event.
519	0207	**JRNotSetup:** The thread is not set up for cond_wait or cond_timed_wait. *Action:* Specify one or more events defined in BPXYCW for the BPX1CWA or BPX1CTW callable services, or use the BPX1CSE callable service prior to BPX1CWA or BPX1CTW.
520	0208	**JRAlreadySetup:** The thread is already set up for cond_setup, cond_wait, or cond_timed_wait. *Action:* Use the BPX1CCA callable service to cancel a condition wait before setting up for a new condition wait.
522	0210	**JRNanoSecondsTooBig:** The value specified for nanoseconds is outside the allowable range. *Action:* Change the value specified for nanoseconds to be less than 1 000 000 000 (1,000 million).
529	0211	**JRTimeOut:** The time for the service to wait has expired. *Action:* While the process was waiting for signals or a condition to occur, the wait time specified expired.
530	0212	**JRDup2Error:** A problem has occurred with the requested file descriptor. *Action:* Try the request again.
546	0222	**JRNoSocket:** The requested operation cannot be performed on a on a socket file descriptor. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
547	0223	**JRMustBeSocket:** The requested operation is valid only on a socket file descriptor. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
549	0225	**JRQuiesceTypeInvalid:** The quiescetype specified by the caller is invalid. *Action:* Reissue the quiesce_threads service with the corrected quiescetype.
550	0226	**JRQuiesceInProgress:** Another thread in the process has already requested quiescing of all threads. *Action:* See your IBM service representative.
551	0227	**JRLastThread:** The last pthread is exiting when the PTFAILIFLASTTHREAD option is specified. *Action:* Reissue the BPX1PTX call without this option to cause the thread to be exited.
552	0228	**JRDomainNotSupported:** The requested domain is not supported. *Action:* The domain must be AF_INET, AF_UNIX, or AF_IUCV.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
553	0229	**JRNetwork:** VFS operation NETWORK is not supported by this file system. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
593	0251	**JROutofSocketsNodeCells:** The system was unable to obtain a cell from the sockets node cell pool. *Action:* See your IBM service representative.
596	0254	**JRSocketNamed:** A bind request was received for a socket that was previously named. *Action:* Do not specify bind for a named socket.
606	025E	**JRSocketCallParmError:** A socket call contains incorrect parameters. *Action:* Correct the parameters and retry the request.
608	0260	**JRInvalidRoutine:** An invalid routine address was passed. *Action:* Reissue the BPX1IPT service specifying a non-zero routine address.
609	0261	**JRRoutineError:** An error occurred while the user provided routine was in control. *Action:* Refer to the provided diagnostic information to resolve the problem.
612	0264	**JRListTooShort:** The read, write, or exception list is too short to contain the specified number of file descriptors and message queue identifiers. *Action:* Reissue the request and specify a larger list.
613	0265	**JRMSOutOfRange:** The value specified for microseconds is outside the allowable range. *Action:* Reissue the request and specify a value for microseconds in the range 0 to 1000000 (one second), inclusive.
614	0266	**JRSecOutOfRange:** The value specified for seconds is outside the allowable range. *Action:* Reissue the request and specify a value for seconds in the range 0 to 2,147,483 (approximately 24.85 days).
617	0269	**JRIncorrectSocketType:** The socket type is incorrect for the request. *Action:* Reissue the request with a different socket type.
617	0269	**JRIncorrectSocketType:** The socket type is incorrect for the request. *Action:* Reissue the request with a different socket type.
626	0272	**JRSocketNotCon:** The requested socket is not connected. *Action:* Make sure the socket is connected and reissue the request.
626	0272	**JRSocketNotCon:** The requested socket is not connected. *Action:* Make sure the socket is connected and reissue the request.
632	0278	**JRSockNoName:** The request requires a socket name structure. *Action:* Specify a socket name.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
636	027C	**JRSockShutDown:** The socket has been shut down. *Action:* The request cannot complete on this socket. Use a different socket.
649	0289	**JRListenNotDone:** The socket is not ready to accept connections. *Action:* Issue a listen (BPX1LSN) request for the socket and then reissue the accept (BPX1ACP) request.
653	028D	**JRListenAlreadyDone:** A listen request has already been completed. *Action:* Issue an accept (BPX1ACP) request to begin accepting connections.
664	0298	**JRECBerror:** The last ECB pointer in the list of ECB pointers does not have the high-order bit (X'80000000') set on to indicate that it is the last ECB pointer in the list. *Action:* Probable user programming error. Ensure that the service was called with the correct number of ECB pointers and that the last ECB pointer has the high-order bit set on.
668	029C	**JRSocketTypeNotSupported:** The requested socket type is not supported. *Action:* Reissue the request with a different socket type.
669	029D	**JREcbWaitBitOn:** The wait (high-order) bit was on in the specified event control block (ECB). *Action:* Initialize the ECB to zero, then reissue the request.
770	0302	**JRIpcBadID:** The ID is not valid or has been removed from the system. *Action:* The specified ID does not represent an active IPC member. Reissue the call with a valid ID.
771	0303	**JRIpcDenied:** Access was denied because the caller does not have the correct permission. *Action:* Access was denied based on the permissions flags set for this IPC member ID on a previous *xxx*get or *xxx*ctl call, and on the effective UID and effective GID of the process. Verify that the correct permissions have been set and that the process is running under the correct effective UID and effective GID. Then reissue the request.
772	0304	**JRIpcExists:** The IPC_CREAT and IPC_EXCL flags were set on the call, and the specified key was already defined to Interprocess Communications. *Action:* The flags indicate that a new member should be created, but a member already exists for the specified key. If you are trying to get the existing member associated with this key, turn off the IPC_EXCL flag and reissue the request. If you are trying to create a new member, reissue the request with a different key.
773	0305	**JRIpcMaxIDs:** The number of IDs exceeds the system limit, and the create failed. *Action:* Remove any IPC members not needed by using the appropriate msgctl, semctl, or shmctl call with the IPC_RMID command. Then reissue the original request.
774	0306	**JRIpcNoExist:** No member exists for the specified key. *Action:* No IPC member is associated with the specified key, and the IPC_CREAT flag is off, indicating that creation of a new member is not allowed. If you are trying to get an existing member, verify that you are using the correct key. Then reissue the request with the correct key. If you are trying to create a new member to be associated with the specified key, turn on the IPC_CREAT flag and reissue the request.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
775	0307	**JRIpcRetry:** NOWAIT was specified, but the operation could not be performed immediately. *Action:* The request would have caused the process to wait for completion, but the IPC_NOWAIT flag specified on the call indicated that the process was not allowed to wait. If the process should wait for the operation to complete, set the IPC_NOWAIT flag off and reissue the request. If the process must never wait, the request can be issued in a loop.
776	0308	**JRIpcSignaled:** An IPC wait was interrupted by a signal. *Action:* The request caused the process to wait (allowed because the IPC_NOWAIT flag was set off), and that wait was interrupted by a signal before the operation could complete. Reissue the request if appropriate.
777	0309	**JRIpcBadFlags:** Extraneous bits were set in the flags parameter or in the mode flag bit field. *Action:* Only those mode flag bits defined for this request may be set on. All other bits must be set off. Verify the bit settings and reissue the request.
778	030A	**JRMsqBadType:** Message type must be greater than zero. *Action:* Use a message type greater than zero and reissue the request.
779	030B	**JRMsqBadSize:** The message length exceeds the system limit or is less than zero. *Action:* Adjust the message length so that it is 0 or greater, but less than the system limit, and reissue the request.
780	030C	**JRMsqNoMsg:** No message of the type requested was found. *Action:* The specified message queue does not contain a message of the desired type, and the IPC_NOWAIT flag was set on. If you expected such a message to exist, verify that the correct *message_queue_ID* and *message_type* were used on the request. If you want to wait for such a message to arrive, reissue the request with the IPC_NOWAIT flag set off.
781	030D	**JRMsq2Big:** The message to receive was too large for the buffer, and the MSG_NOERROR flag was not specified. *Action:* The requested message is too large to fit within the requested length (as specified by the *message_size*) parameter. The MSG_NOERROR flag was set off, which did not allow the message to be truncated to fit within that length. If truncation of the message is desired, set MSG_NOERROR on and reissue the request. If the entire message is desired, increase the size of the buffer and the *message_size* parameter accordingly and reissue the request.
782	030E	**JRSema4BadAdj:** The value specified would exceed the system limit for *semadj*. *Action:* The operation would cause the *semval* or *semadj* value to overflow the system-imposed limit defined in the BPXYSEM macro. Adjust the operation and reissue the request.
783	030F	**JRSema4BadNOps:** The specified number of semaphore operations exceeds the system limit. *Action:* Decrease the number of semaphore operations requested and reissue the request.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
784	0310	**JRSema4BadNSems:** A semaphore ID exists for the specified key, but the number of semaphores is not valid. *Action:* A semaphore set ID exists for the specified key, but the number of semaphores requested exceeds the number of semaphores that were defined when this semaphore set was created. Adjust the number of semaphores on this request to be less than or equal to the maximum number of semaphores in the set and reissue the request.
786	0312	**JRSema4BadSemN:** The semaphore number is not valid. *Action:* The specified semaphore number is less than zero or greater than the number of semaphores in the set. Correct the semaphore number to fall within these limits and reissue the request.
787	0313	**JRSema4BadValue:** The value specified would exceed the system limit. *Action:* The value of *semval* specified in the *value_or_address* parameter or in the array pointed to by the *value_or_address* parameter exceeds the system-imposed maximum defined in the BPXYSEM macro. Correct the value and reissue the request.
788	0314	**JRSema4BigNSems:** The number of semaphores exceeds the system maximum. *Action:* The number of semaphores requested to be allocated to the set exceeds the system-defined limit. Correct the value and reissue the request.
789	0315	**JRSema4ZeroNSems:** The number of semaphores requested was zero, and the semaphore set does not exist. *Action:* Specifying zero as the number of semaphores is allowed only if the key is associated with an existing semaphore set. However, the specified key is not associated with any existing semaphore set. If you are trying to get an existing semaphore set, verify that you are using the correct key and reissue the request with the correct key. If you want to create a new semaphore set to be associated with this key, specify the number of semaphores to be defined for the set and reissue the request.
790	0316	**JRShmBadSize:** The shared memory segment size is incorrect or outside the system-defined range of valid segment sizes. *Action:* The requested shared memory size for the existing shared memory segment associated with the specified key cannot be greater than the shared memory size that was defined when the shared memory segment was created. Verify that the correct key was specified. If so, adjust the requested shared memory size appropriately and reissue the request.
791	0317	**JRShmMaxAttach:** The number of shared memory segments attached for the current process exceeds the system-defined maximum. *Action:* Use shmdt (BPX1MDT) to detach some shared memory segments and then reissue the request.
792	0318	**JRIpcRemoved:** During a wait, the IPC member ID was removed from the system. *Action:* A request caused the process to wait (allowed by the IPC_NOWAIT flag being set off), and during that wait the IPC member was removed from the system. This IPC member is no longer available.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
793	0319	**JRMsqQBytes:** Not permitted to increase the MSG_QBYTES value, or an attempt by a superuser to set the MSG_QBYTES exceeds the system limit. *Action:* You must be a superuser to issue a msgctl (BPX1QCT) request to increase the number of bytes allowed on a queue (MSG_QBYTES), and that value must not exceed the system-defined maximum.
796	031C	**JRMsqQueueFullMessages:** IPC_NOWAIT was specified, but the operation was not done because there was no room in the message queue due to the number of messages in the message queue. *Action:* Use the msgrcv (BPX1QRC) service to receive some messages off the queue, or set the IPC_NOWAIT flag off to wait for room on the queue, and reissue the request.
797	031D	**JRMsqQueueFullBytes:** IPC_NOWAIT was specified, but the operation was not done because there was no room in the message queue due to the number of bytes in the message queue. *Action:* Use the msgrcv (BPX1QRC) service to receive some messages off the queue, or set the IPC_NOWAIT flag off to wait for room on the queue, or have the MSG_QBYTES limit increased (by a superuser), and reissue the request.
799	031F	**JRSemStorageLimit:** The semget or semop call failed because the semaphore storage limit was reached. *Action:* Release some system storage by cleaning up unneeded resources within the application or outside the application, and then reissue the request.
804	0324	**JRSmNoStorage:** There is no storage available to allocate. *Action:* Release some system storage by cleaning up unneeded resources within the application or outside the application, and then reissue the request.
829	033D	**JRInvalidResource:** The input resource value is not valid. *Action:* Reissue the request with a valid resource value.
837	0345	**JRPathconf:** The pathconf service vnode operation is not supported. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
870	0366	**JRWriteBeyondLimit:** Cannot write beyond the file size limit. *Action:* Write less data to the file.
880	0370	**JRBadIDType:** The ID type passed on the call was not valid. *Action:* Reissue the request with a valid ID type.
881	0371	**JRBadOptions:** The options parameter contained options that were not valid. *Action:* Reissue the request with valid options.
897	0381	**JRPrevSockError:** A previous error caused this socket to become unusable. *Action:* Close the socket.
926	039E	**JRTooManyFds:** Too many file descriptors were specified. *Action:* Reduce the number of Fds specified to a number that is supported by the service.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
942	03AE	**JRBatSel:** The batch-select VFS operation is not supported. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
945	03B1	**JRCMSLoadFailure:** A call to the DLL_load (BPX1LOD) service caused a failure in the CMS LOADMOD routine. *Action:* Make sure that the specified file is a relocatable, executable CMS MODULE created by the GENMOD command, the BIND command, the c89 utility, or the cxx utility.
953	03B9	**JRWaitForever:** A timeout pointer value of 0 (wait forever) was specified, but there were no events to wait for. *Action:* Reissue the request and specify at least one event or change the timeout pointer value to point to a timeout value.
954	03BA	**JRInvalidNfds:** The NFDS parameter was larger than the number of open files for the process. *Action:* Reissue the request, specifying a value for the NFDS parameter that is less than DFLT_OPEN_MAX.
961	03C1	**JRNoFdsTooManyQIds:** The number of Fds specified was negative, or too many message queue IDs were specified on the select service. *Action:* Reissue the select request, specifying a non-negative number of Fds, or reduce the number of message queue IDs to be processed to below the maximum supported by the system.
978	03D2	**JRBadID:** An incorrect ID value was passed to the wait_extension (BPX1WTE) service. *Action:* Reissue the call with a valid ID.
1026	0402	**JRCancel:** Vnode operation CANCEL is not supported by this file system. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
1027	0403	**JRDuplicateCancel:** A cancel operation is already in progress for the target asyncio request. *Action:* Wait for the previous cancel request to complete.
1045	0415	**JRAsyncOpNotSupp:** The asyncio operation cannot be performed because the socket transport does not support asynchronous I/O, or asyncio select included a PFS that could not support this operation. *Action:* Verify that the operation was performed on a physical file system that supports the operation.
1123	0463	**JRAnr:** Vnode operation ACCEPT_AND_RECEIVE is not supported by this file system. *Action:* Issue separate accept and receive operations with this socket.
1124	0464	**JRSrx:** Vnode operation SR_CSM is not supported by this file system. *Action:* Use regular send and receive operations with this socket.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
1500	05DC	**JREcbError2:** If an ECB pointer is used on selectex, the timeout pointer must be 0 (wait forever). *Action:* Correct the inputs and reissue the request.
1503	05DF	**JRTransportError:** The sockets transport layer returned the error. For AF_INET sockets, the transport layer is the TCP/IP stack. For AF_IUCV and AF_UNIX, the transport layer is IUCV. *Action:* Refer to the return code for information on the error.
1504	05E0	**JRIPv6NotEnabled:** IPv6 is not enabled on the TCP/IP stack. *Action:* Ensure you are using the correct stack (check TCPIPUSERID entry of TCPIP DATA file) and ensure that the stack is correctly configured for IPv6.
2000	07D0	**JRCPNotFound:** Name or ID is not found. *Action:* Reissue the request specifying a valid user ID, user name, group Id, or group name.
2001	07D1	**JRCPNotAuthorized:** Not authorized for search. *Action:* Contact your system administrator to obtain the proper authority to issue the request to query the user data base.
2002	07D2	**JRCPNotAvail:** User data base not available. *Action:* Contact your system administrator to find out the status of the user data base.
2006	07D6	**JRCPInternalError:** Internal CP/CMS error. *Action:* See your IBM service representative.
2007	07D7	**JRCPUserNotFound:** User not found. *Action:* Reissue the request specifying a valid user name.
2008	07D8	**JRIdentifyErr:** Call to Identify failed. *Action:* See your IBM service representative.
2009	07D9	**JRStackReadErr:** Call to StackRead failed. *Action:* See your IBM service representative.
2010	07DA	**JRQEFLErr:** Call to DMSQEFL failed. *Action:* See your IBM service representative.
2011	07DB	**JRInvFilePoolID:** The filepool identifier in the fully-qualified path name is not valid. *Action:* Reissue the request specifying a valid file pool id.
2012	07DC	**JRInvFileSpaceID:** The file space identifier in the fully-qualified path name is not valid. *Action:* Reissue the request specifying a valid file space id.
2013	07DD	**JRNoMoreVFSs:** All Virtual File System (VFS) control blocks in the FSSM are allocated. *Action:* The maximum number of concurrent mounts has been reached. Issue OPENVM UNMOUNT for any unneeded mount points.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
2014	07DE	**JRNoMoreMtabs:** All Mount Table Entry (MTAB) control blocks in the FSSM are allocated. *Action:* The maximum number of concurrent mounts has been reached. Issue OPENVM UNMOUNT for any unneeded mount points.
2015	07DF	**JRNoMoreVnods:** All VNODEs in the FSSM are allocated. *Action:* Try to free references to active objects by closing files or directories and reissue the request. Or, logoff or reIPL CMS.
2018	07E2	**JRCtyConnectionInop:** The Cty connection is inoperative. *Action:* An incorrect action code was specified. Correct the program and rerun.
2019	07E3	**JRCtyInvalidAction:** The action code is incorrect. *Action:* See your IBM service representative.
2020	07E4	**JRCtyNoCntlTerm:** The caller has no controlling terminal. *Action:* The caller has no controlling terminal. Correct the program or rerun in an environment where the file is for the controlling terminal.
2021	07E5	**JRCtyDiffSession:** This is not the callers controlling terminal. *Action:* The specified file descriptor is not for the callers controlling terminal. Correct the program or rerun in an environment where the file is for the controlling terminal.
2022	07E6	**JRCtyInvalidPgid:** The requested process group ID is not valid. *Action:* The specified process group ID is not a valid OpenVM process group ID. Correct the program and rerun.
2023	07E7	**JRCtyNotInSession:** The process group ID (PGID) does not exist in the callers session. *Action:* The callers session does not have a process group with the specified process group ID. The process group may have already completed. Correct the program and rerun.
2024	07E8	**JRCtyNotPGLeader:** The process is not a process group leader. *Action:* The specified process group ID does not represent a process group leader. Correct the program or rerun in an environment where the process is a process group leader.
2025	07E9	**JRCtyBgCall:** This is a background process. *Action:* The service requested is not allowed from the background. Rerun the program in the foreground.
2026	07EA	**JRCtyBadQueSel:** The queue selector is not valid. *Action:* An incorrect queue selector was specified. Correct the program and rerun.
2027	07EB	**JRCtyOrphanedWrite:** The write service is processing in a background orphaned process group. *Action:* This condition most likely occurs when a process that is spawned from the session leader attempts to write to the terminal after the session leader process has ended. The process cannot read from or write to the terminal once the session leader process ends. This terminal session is no longer usable. Restart the application from another session.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
2028	07EC	**JRCtySIGTTINBlocked:** The process is in a background process group and SIGTTIN is blocked or ignored. *Action:* If the SIGTTIN signal is either blocked or ignored, the read call can be issued only from a process that is running in a foreground process group.
2029	07ED	**JRCtyInputStopped:** Nonblocked read failed, because input is stopped. *Action:* Input has been stopped by a tcflow service. Issue a tcflow to start input, and reissue the read.
2030	07EE	**JRCtyOutputStopped:** Nonblocked write failed, because output is stopped. *Action:* Output has been stopped by a tcflow service. Issue a tcflow to start output, and reissue the write.
2031	07EF	**JRCtyOrphanedRead:** The read service is processing in a background orphaned process group. *Action:* This condition most likely occurs when a process that is spawned from the session leader attempts to write to the terminal after the session leader process has ended. The process cannot read from or write to the terminal once the session leader process ends. This terminal session is no longer usable. Restart the application from another session.
2032	07F0	**JRCtyNoData:** Data or room is not available on the queue. *Action:* Non-blocking read was issued, but there is no data on the input queue. Reissue the request again later.
2033	07F1	**JRCtyDeviceError:** I/O error occurred during terminal read or write. *Action:* An I/O error occurred when process was trying to read from or write to a terminal. Reissue the request again.
2034	07F2	**JRCtyAlreadyActive:** The process has already opened a terminal file. *Action:* An attempt was made to open another terminal file. Close an opened terminal, then retry.
2035	07F3	**JRInvDeviceId:** The fully-qualified root contains the reserved POSIX device ID, but the qualifying path name is unsupported or invalid. *Action:* Reissue the request specifying a valid path name.
2036	07F4	**JRLinkNotFound:** The data associated with the external link cannot be found. *Action:* Use the OPENVM QUERY LINK command to verify the external link. Also, check your search order to make sure you have the necessary directories and minidisks accessed.
2049	0801	**JRBindBadState:** FSSTATE macro returned an unexpected return code while checking the existence of a CMS module file on a CMS minidisk. *Action:* Message DMS1745S contains the FSSTATE return code. More information on FSSTATE can be found in *z/VM: CMS Macros and Functions Reference*.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
2050	0802	**JRBindNotOpenedI:** FSOPEN failed to open the CMS module file for input. The CMS Binder checks an existing module to determine if it is executable before replacing it with a non-executable extended format CMS module. *Action:* Message DMS1262S contains the FSOPEN return code. More information on FSOPEN can be found in *z/VM: CMS Macros and Functions Reference*.
2051	0803	**JRBindNotOpenedO:** FSOPEN failed to open the CMS module file for output. The CMS Binder is unable to open the required CMS module to write the new CMS module file. *Action:* Message DMS1262S contains the FSOPEN return code. More information on FSOPEN can be found in *z/VM: CMS Macros and Functions Reference*.
2051	0803	**JRChmodFiletype:** Mode (file type) mismatch on chmod. *Action:* Reissue the request specifying a valid mode.
2052	0804	**JRBindNotClosed:** FSCLOSE failed to close the CMS module file. *Action:* Message DMS1262S or DMS1740E contains the FSCLOSE return code. More information on FSCLOSE can be found in *z/VM: CMS Macros and Functions Reference*.
2052	0804	**JRInvalidAttr:** Invalid Attr input to vnode operation. *Action:* See your IBM service representative.
2053	0805	**JRBindBadRead:** The FSREAD macro returned an unexpected return code while reading a record from a CMS module file on a CMS minidisk. *Action:* Message DMS104S contains the FSREAD return code. More information on FSREAD can be found in *z/VM: CMS Macros and Functions Reference*.
2053	0805	**JRInvalidCjar:** Invalid Cjar input to vnode operation. *Action:* See your IBM service representative.
2054	0806	**JRBindBadWrite:** FSWRITE macro returned an unexpected return code while writing a record to a CMS module file on a CMS minidisk. *Action:* Message DMS105S contains the FSWRITE return code. More information on FSWRITE can be found in *z/VM: CMS Macros and Functions Reference*.
2055	0807	**JRInvalidFileType:** Invalid file type for current operation. *Action:* Reissue the request specifying a valid type of file.
2056	0808	**JRInvalidForSymlink:** This operation is invalid for symbolic links. *Action:* Reissue the request specifying a symbolic link.
2057	0809	**JRInvalidMtab:** Invalid mount table entry. *Action:* See your IBM service representative.
2058	080A	**JRInvalidIName:** Input name (*terminal pathname component*) invalid. *Action:* See your IBM service representative.
2059	080B	**JRInvalidToken:** Invalid Token Manager token. *Action:* See your IBM service representative.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
2060	080C	**JRInvalidUIO:** Invalid UIO input to vnode operation. *Action:* See your IBM service representative.
2061	080D	**JRLockRetryLim:** Number of lock retries exceeded; reissue request. *Action:* This should be a transient condition. Retry the service. If the problem persists, contact the system programmer or system administrator to diagnose the problem.
2062	080E	**JRNotBFS:** Not a Byte File Space. *Action:* Reissue the request, specifying the name of a Byte File System.
2063	080F	**JRObjectInUse:** Byte-Range/Object/Directory/File Space/Storage Group is in use. *Action:* A lock or object token conflict was detected that may involve extensive delay. Issue the request again. If it still fails, determine if the object is locked explicitly using QUERY FILEPOOL DISABLE or QUERY LOCK. Or, in the case where the conflict is for an object token and reissuing the request does not resolve this, your system administrator should determine if a client virtual machine is in a loop or other condition where it does not respond.
2064	0810	**JRBindNoStorage:** CMSSTOR macro returned an unexpected return code while attempting to obtain storage to generate a CMS module file on a CMS minidisk. *Action:* Refer to the associated messages issued by CMSSTOR for the CMSSTOR return code and error description. More information on CMSSTOR can be found in *z/VM: CMS Macros and Functions Reference*.
2064	0810	**JRRemoveTopDir:** Cannot remove top directory. *Action:* The BFS top directory cannot be removed with this request. Ask your file pool administrator to issue the DELETE USER command or DMSDEUSR CSL routine to remove the file system.
2065	0811	**JRSoftLinkError:** Soft link creation error. *Action:* See your IBM service representative.
2066	0812	**JRStorageObtainErr:** Error obtaining free storage. *Action:* You must either free some virtual storage or increase the size of your virtual machine. To increase the size of your virtual machine, use the DEFINE command; then reIPL CMS and enter the original command again.
2067	0813	**JRStorageReleaseErr:** Error releasing free storage. *Action:* ReIPL and reissue the command. If the problem persi ensure that the application you are using is not corrupting storage. If that doesn't help, contact system support personnel to correct the problem.
2071	0817	**JRInvalidAuthStruc:** Invalid authorization structure on MakeCatRow. *Action:* See your IBM service representative.
2072	0818	**JRInvalidInputBuf:** Invalid input buffer on MakeCatRow. *Action:* See your IBM service representative.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
2073	0819	**JRInvalidOutputBuf:** Invalid output buffer on MakeCatRow. *Action:* See your IBM service representative.
2074	081A	**JRUDCError1:** User Data Cache error (buffer header management). *Action:* See your IBM service representative.
2075	081B	**JRUDCError2:** User Data Cache error (block header management). *Action:* See your IBM service representative.
2076	081C	**JRUDCError3:** User Data Cache error (LRU queue management). *Action:* See your IBM service representative.
2077	081D	**JRNoRecall:** File is migrated, but RECALL is off. *Action:* SET RECALL ON and enter the request again, or enter the DFSMS RECALL command to explicitly recall the file.
2078	081E	**JRFsNotRegFile:** The fsync() call is supported only for regular files. *Action:* Reissue the request specifying the name of a regular file.
2079	081F	**JRNothingMounted:** The umount service was issued, but nothing was mounted. *Action:* The requested service cannot be processed.
2080	0820	**JRBindNegativeLength:** A negative length for the linear program object was passed in the parameter list to the BPX1WRT CMS Binder interface *Action:* There is a program logic error in CMS Binder processing. Collect any relevant information and report the problem to your IBM service representative.
2080	0820	**JRGetFQName:** The GetFQName vnode operation is not supported. *Action:* See your IBM service representative.
2081	0821	**JRDuplicateMEL:** There can only be 1 reference to a target of a mount external link (MEL) at a time. *Action:* Unlink the original MEL or reIPL CMS and retry the request.
2082	0822	**JRFileSpaceUnknown:** The specified file spaces does not exist, or is not a BFS file space. *Action:* Reissue the request specifying a valid BFS file space id.
2083	0823	**JRMountNotFQName:** The file system to be mounted must be a fully-qualified path name. *Action:* Reissue the request specifying a fully-qualified path name.
2084	0824	**JRNoExtLink:** The requested operation cannot be performed on an external link. *Action:* Reissue the request specifying a file that is not an external link.
2095	082F	**JRExtFileDoesNotExist:** The CMS file referenced by the specified external link does not exist. *Action:* An external link of subtype FST_EXEC or subtype FST_DATA was created, and a command was entered against the external link that required the CMS file referenced by the external link to exist. Reissue the request, specifying a link name that references an existing file.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
2096	0830	**JRExtFileAlreadyExists:** The file requested for creation as an external link already exists. *Action:* The link name specified on an extlink service request is an existing file name. Reissue the request specifying a link name that does not already exist.
2097	0831	**JRBindBadPOdata:** A program object does not contain the required eyecatcher *Action:* If you are trying to replace an existing extended format CMS module file on the output CMS minidisk, delete it and retry the operation. If the problem still persists, collect any relevant information and report the problem to your IBM service representative.
2097	0831	**JRExtlink:** The external link vnode operation is not supported. *Action:* See your IBM service representative.
2098	0832	**JRBindDuplicatePOKey:** A duplicate key has been detected in the linear program object header. *Action:* There is a program logic error in the CMS Binder processing. Collect any relevant information and report the problem to your IBM service representative.
2098	0832	**JRFileNotExtLink:** The file requested for readlink service is not an external link. *Action:* Reissue the readlink service request specifying the name of an external link.
2099	0833	**JRBindInvalidPOKey:** An invalid key has been detected in the linear program object header. *Action:* There is a program logic error in the CMS Binder processing. Collect any relevant information and report the problem to your IBM service representative.
2099	0833	**JRInvalidExtLinkLen:** The contents specified for an external link has an incorrect length. *Action:* Reissue the extlink request specifying a path length greater than or equal to zero, but less than 1023.
2100	0834	**JRBindMissingPOKey:** A required key field is missing from the linear program object header. *Action:* There is a program logic error in CMS Binder processing. Collect any relevant information and report the problem to your IBM service representative.
2100	0834	**JREndingSlashExtlink:** The path name ended with slash on the extlink service. *Action:* The link name specified on an extlink request contained a trailing slash. Reissue the request omitting the trailing slash.
2101	0835	**JRNoMorePNEs:** All path name cache entries are taken. *Action:* There are no free path name cache entries in the FSSM. Logoff or reIPL CMS.
2102	0836	**JRInvCWD:** The working directory as defined cannot be resolved. *Action:* Examine your current working directory using the OPENVM QUERY DIRECTORY command to ensure that the path name is qualified the way you want it to be.
2103	0837	**JRInvRoot:** The root directory as defined cannot be resolved. *Action:* Examine your file system root using the OPENVM QUERY MOUNT command to ensure that the path name is qualified the way you want it to be.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
2104	0838	**JRStorageGroupFull:** The storage group containing the file space is full. *Action:* Remove some regular files from your file space or ask the file pool administrator to issue the MODIFY USER command to increase the size of your file space.
2105	0839	**JRFileSpaceFull:** The file space is full. *Action:* Remove some regular files from your file space or ask the file pool administrator to issue the FILEPOOL MINIDISK command to increase the size of the storage group.
2106	083A	**JRNoDFSMS:** DFSMS not active or SEND or RECALL exit not active. *Action:* Contact your file pool administrator to determine why DFSMS is not available for your file pool.
2107	083B	**JRBRMCancel:** Canceled by the BRM cancel request. *Action:* Retry the request or contact your system administrator to determine why your request was canceled.
2108	083C	**JRPipeProcErr:** Generic pipe processor errors. *Action:* See your IBM service representative.
2110	083E	**JRRenameTopDir:** Cannot rename top directory. *Action:* The BFS top directory cannot be renamed with this request. Ask your file pool administrator to issue the FILEPOOL RENAME command to rename the file system.
2111	083F	**JRSerStorageObtainErr:** Error obtaining free storage in file pool server. *Action:* You must either increase the size of the file pool server virtual machine or use a different file pool server. Contact your file pool administrator and inform him of the problem.
2112	0840	**JRBindDuplicateModule:** A CMS module with the same file name already exists on the output disk and the REPLACE=NO option has been specified. *Action:* Refer to message DMS1905S for the output module name. Specify the REPLACE option on the name statement or SAVEW API parameter list and retry the operation to replace the module.
2112	0840	**JRMaxconnExceeded:** APPC/VM maxconn exceeded for this virtual machine. *Action:* If your user ID limit was reached, logoff or reIPL CMS to remove the existing connections. Or contact the administrator ofr the CP directory entry for your user ID to increase the MAXCONN value.
2113	0841	**JRBindNXstdModule:** The generated linear program object is non-executable and cannot be stored as a standard format CMS module. This only occurs when either a COMPAT(PM1) or COMPAT(LKED) option has been specified. *Action:* Refer to the associated CMS Binder messages to determine why the module is non-executable and either remedy the reported error or specify a LET option value so the program module is not marked non- executable. Otherwise, specify either COMPAT(PM2) or COMPAT(PM3) to generate an extended format CMS module, which can be non-executable.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
2113	0841	**JRFilePoolUnknown:** Unknown or unavailable file pool. *Action:* The file pool ID is part of the the fully-qualified path name. If you are not using a fully-qualified path name, use OPENVM QUERY MOUNT to determine your root and the name of the file pool. Once you have determined the file pool ID used, contact your system support personnel to determine the status of the file pool.
2114	0842	**JRBindNoStoreNXstd:** The generated linear program Object is non-executable and cannot replace an existing standard CMS module on the same output CMS minidisk. *Action:* Refer to the associated CMS Binder messages to determine why the module is non-executable and either remedy the reported error or specify a LET option value so the program module is not marked non- executable. Otherwise, specify the STORENX=YES option and retry the operation to permit the non-executable module to replace the existing module file.
2114	0842	**JRFilePoolSever:** Connection to file pool has been severed. *Action:* Notify system support personnel or the file pool administrator that the file pool is unavailable.
2115	0843	**JRBindNoStoreNXext:** The generated linear program object is non-executable and cannot replace an existing executable extended format CMS module on the same output CMS minidisk. *Action:* Refer to the associated CMS Binder messages to determine why the module is non-executable and either remedy the reported error or specify a LET option value so the program module is not marked non- executable. Otherwise, specify the STORENX=YES option and retry the operation to permit the non-executable module to replace the existing module file.
2115	0843	**JRSvrMaxconnExceeded:** APPC/VM maxconn exceeded for file pool server. *Action:* Contact the file pool administrator. The administrator should either increase the MAXCONN value for the server machine or somehow decrease the number of users accessing the file pool at any one time.
2116	0844	**JRNoMoreIOCache:** I/O cache (user data cache) buffers are all in use. *Action:* See your IBM service representative.
2117	0845	**JRConnectAuthFailure:** Not authorized to connect to file pool. *Action:* Contact the file pool administrator. The administrator should either enroll you by name into the file pool, ENROLL PUBLIC for the file pool, or assign a POSIXINFO UID statement to your CP directory entry.
2118	0846	**JRNFSCBranchFail:** LFS error communicating with NFS client. *Action:* See your IBM service representative.
2119	0847	**JRNFSCInitFail:** General NFS Client initialization failure. *Action:* See your IBM Service Representative.
2120	0848	**JRNFSCInitFail1:** SCREERUN load error during NFS Client LFS session initialization. *Action:* See your IBM service representative.
2121	0847	**JRNFSCInitFail2:** NFS MODULE load error during NFS Client LFS session initialization. *Action:* See your IBM service representative.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
2122	084A	**JRNFSCInitFail3:** CMS MT init error during NFS Client LFS session initialization. *Action:* See your IBM service representative.
2123	084B	**JRNFSCInitFail4:** Queue failure during NFS Client LFS session initialization. *Action:* See your IBM service representative.
2124	084C	**JRNFSCInitFail5:** Root thread failure during NFS Client LFS session initialization. *Action:* See your IBM service representative.
2125	084D	**JRNFSCInitFail6:** Queue failure during NFS Client process initialization. *Action:* See your IBM Service Representative.
2126	084E	**JRNFSCInitFail7:** Queue failure during NFS Client local initialization. *Action:* See your IBM Service Representative.
2127	084F	**JRNFSCInitFail8:** PFS init failure during NFS Client local initialization. *Action:* See your IBM service representative.
2128	0850	**JRBindPSGMUnsupported:** The generated linear program object contains unsupported overlay segment information. *Action:* Overlay segments are not supported by CMS. Modify the source program to remove the overlay segments. Then recompile and BIND to generate a module that does not use overlay segments.
2128	0850	**JRNFSCReqFail:** General failure during NFS request. *Action:* See your IBM service representative.
2129	0851	**JRBindBadLIDXsegment:** A loader data segment that is not valid has been detected in the linear program object while building a standard format CMS module. *Action:* Collect any relevant information and report the problem to your IBM service representative.
2129	0851	**JRNFSCReqFail1:** Queue failure during NFS Client request processing. *Action:* See your IBM Service Representative.
2130	0852	**JRBindBadSegmentId:** An segment ID that is not valid was detected in the relocation data of the linear program object while building a standard format CMS module. *Action:* To circumvent, specify either COMPAT(PM2) or COMPAT(PM3) to build an extended format CMS module. Otherwise, collect any relevant information and report the problem to your IBM service representative.
2130	08520	**JRNFSCReqFail2:** Thread create failure during NFS request processing. *Action:* See your IBM Service Representative.
2131	0853	**JRBindBadRDTFormat:** Relocation data contains an incorrect or unsupported format. *Action:* To circumvent, specify either COMPAT(PM2) or COMPAT(PM3) to build an extended format CMS module. Otherwise, collect any relevant information and report the problem to your IBM service representative.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
2131	0853	**JRNFSCMntTCPIPDATA:** TCPIP DATA file not found. *Action:* Access the disk containing the TCPIP DATA file.
2132	0854	**JRBindRDT1outsideText:** A format 1 relocation data entry contains an incorrect relocation offset. *Action:* To circumvent, specify either COMPAT(PM2) or COMPAT(PM3) to build an extended format CMS module. Otherwise, collect any relevant information and report the problem to your IBM service representative.
2132	0854	**JRNFSCMntTCPXLBIN:** POSIX TCPXLBIN or other TCPXLBIN file not found. *Action:* Access the disk containing the TCPXLBIN file specified in the mount operation.
2133	0855	**JRBindRDT2outsideText:** A format 2 relocation data entry contains an incorrect relocation offset. *Action:* To circumvent, specify either COMPAT(PM2) or COMPAT(PM3) to build an extended format CMS module. Otherwise, collect any relevant information and report the problem to your IBM service representative.
2133	0855	**JRNFSCInitFail9:** Error getting socket during NFS client init.. *Action:* See your IBM Service Representative.
2134	0856	**JRBindRDT3outsideText:** A format 3 relocation data entry contains an incorrect relocation offset. *Action:* To circumvent, specify either COMPAT(PM2) or COMPAT(PM3) to build an extended format CMS module. Otherwise, collect any relevant information and report the problem to your IBM service representative.
2134	0856	**JRNFSCInitFail10:** Queue failure during NFS Client local init. *Action:* See your IBM Service Representative.
2135	0857	**JRBindRDT3BadFormat:** The processing of a format 3 relocation data entry has detected incorrect relocation offset. *Action:* To circumvent, specify either COMPAT(PM2) or COMPAT(PM3) to build an extended format CMS module. Otherwise, collect any relevant information and report the problem to your IBM service representative.
2135	0857	**JRPfsCtl:** PFSCTL function not supported by file system. *Action:* See your IBM service representative.
2136	0858	**JRBindBad24BitAddress:** Processing a 24 bit relocation data entry has generated an address greater than X'00FFFFFF'. *Action:* Revise the program structure to utilize 31 bit programming techniques.
2136	0858	**JRNFSC2ManyRestart:** Too many restart attempts have been made in error. *Action:* See your IBM Service Representative.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
2137	0859	**JRBindAliasUnsupported:** Alias Information has been detected while building either a standard or extended format CMS module on a CMS minidisk. Alias information is not supported in this environment. *Action:* Remove ALIAS statements from the CMS Binder input commands and statements. Then retry the CMS BIND operation.
2137	0859	**JRNFSError:** Error reported by NFS client. *Action:* Examine the return code to determine the reason for the error.
2138	085A	**JRNFSNamNotAllowed:** Error fully-qualified name not allowed. *Action:* Retry the operation specifying a name that does not represent an object in any NFS-mounted file system.
2139	085B	**JRGetHostNameError:** Error on Gethostname call. *Action:* Check the host name specification in the TCPIP DATA file.
2140	085C	**JRPCNFSNotAvail:** PCNFS protocol not available at server. *Action:* Retry the request without specifying user ID.
2141	085D	**JRPCNFSError:** Error calling PCNFS. *Action:* Retry the request without specifying user ID.
2142	085E	**JRBadNFSpathname:** Invalid NFS pathname. *Action:* Retry the request specifying a fully-qualified NFS path name in the correct format.
2143	085F	**JRBadTCPIPDATA:** Error using TCPIP DATA file. *Action:* Determine why the TCPIP DATA file cannot be read.
2144	0860	**JRBadNETRCFile:** Error using NETRC DATA file. *Action:* Determine why the NETRC DATA file cannot be read.
2145	0861	**JRNETRCFileNotFound:** NETRC DATA file not found. *Action:* Create an NETRC DATA file, or specify user ID and password, or ANONYMOUS, on the mount request.
2146	0862	**JRMntNoPassword:** User ID provided, but no password. *Action:* Specify a password on the mount request or in the NETRC DATA file.
2147	0863	**JRStatVFS:** StatVFS function not supported by file system. *Action:* None.
2148	0864	**JRNFSVerNotSupp:** Requested version of NFS not supported. *Action:* Retry the mount request specifying a different version.
2149	0865	**JRNFSDecodeFail:** NFS server option is not valid. *Action:* Correct the NFS server options specified as part of the Network File System path name and retry the request.

Table 8. OpenExtensions Reason Codes by Numeric Value (continued)

Dec Value	Hex Value	Description
2150	0866	**JRBadTCPXLBIN:** Error using TCPXLBIN file. *Action:* Determine why the TCPXLBIN file cannot be read.
2151	0867	**JRNFSNotAllowed:** Operation not allowed for an object in an NFS-mounted file system. *Action:* Retry the request specifying a path name that does not represent an object in an NFS-mounted file system.
2152	0868	**JRGIDLimitExceeded:** GID Supplemental List limit exceeded. *Action:* Decrease the number of entries in the supplemental GID list.
2154	086A	**JRNFSCSocketFailed:** Failure on a socket being used to communicate with a remote NFS server. *Action:* Retry the request.
2155	086B	**JRNFSCNoPermMount** The export list at the remote NFS server does not contain an entry that allows you to mount the directory, you do not have permission for the directory, or the NFS server requires that the NFS client use a low port number (in the range 0 to 1023). *Action:* Contact the system administrator for the remote host to ask that the export list be updated. If the NFS server allows mounting of non-exported file systems, contact the owner of the file system to update permissions. If the NFS server allows mounting of only exported systems and the export list contains an entry allowing you to mount, it may be that the NFS server requires the use of a low port number (in the range 0 to 1023). Contact the system administrator for the remote host to ask that the remote NFS server configuration be changed to permit clients to use any port number. The system administrator should consult the NFS server documentation to determine how this is done. The documentation may make reference to "secure" or "insecure" port numbers.
2156	086C	**JRNFSMountError** Error during NFS mount. *Action:* Use the OPENVM FORMAT command to display information about why the mount attempt failed.
2157	086D	**JRPortMapperError** Error calling port mapper. *Action:* Contact the TCP/IP administrator for the foreign host to determine why the port mapper function is not available.
2158	086E	**JRConnectSSIFailure** The only connections to the file pool that are allowed are those from within the SSI cluster. *Action:* Retry connecting to the file pool from within the SSI cluster.

Special CMS File Pool Server and BFS Client Reason Codes

The following is a list of file pool server internal reason codes. They may be displayed when the return code is X'A2' and the reason code qualifier is in the range X'5B00' to X'5BFF'. (Reason code qualifier X'5B01' is for BFS client internal reason codes, which are not listed here.) All of these codes represent a system error for which you should see your IBM service representative.

Table 9. File pool server internal reason codes

Dec Value	Hex Value	Description
200	0C8	Object already exists or duplicate key violation in catalog
450	1C2	Invalid input type for BFS server request
466	1D2	File space is not a BFS file space
469	1D5	Invalid token passed in BRMIN. The token could not be resolved to a BFCB (by read, write, or close) or to a DCB (by readdir or closedir).
470	1D6	File not open for write (detected by write) or proper token not held
471	1D7	No file space found (no SPACECAT row)
474	1DA	Invalid number of buffers passed in the BRMIN or invalid filesize (for write).
478	1DE	Attempted to unlink a directory or attempted to open a directory using a BRM OPEN request or attempted to do a pipe open for an object that is not a named pipe.
484	1E4	Invalid length on pipe read (< 0 | > 16K)
485	1E5	Mismatch in code levels among the server modules.
499	1F3	Overflow in the HIGHINO or HIGHNID
561	231	Required lock not in effect.
713	2C9	An unexpected error was returned on the call to SAC lock functions.
716	2CC	An unexpected error was returned on a call to an SC Function
729	2D9	A call was attempted for an unsupported request.
730	2DA	System error in DAC. This could occur when DAC modules detect input parameters that are no supposed to occur or when control blocks contain data that's inconsistent or should not be there.
733	2DD	System error in file pool server data access component.
734	2DE	Sliver condition in BRM Space Management.
740	2E4	Inconsistent Catalogs.
760	2F8	File pool server commit processing error. Commit is not done and an implicit rollback is performed.
790	316	Invalid input - should not occur situations
791	317	Search key does not resolve to an object (internal token mgr error)
792	318	Invalid object pointer (internal token mgr error)
972	3CC	System error in Byte File Linkage Component

OpenExtensions Reason Codes Listed by Symbolic Name

Table 10. OpenExtensions Reason Codes by Symbolic Name

Reason Code	Decimal	Hex
JRAccess	331	014B
JRAllFilesNotClosed	470	01D6
JRAlreadyDetached	475	01DB
JRAlreadyJoined	480	01E0
JRAlreadySetup	520	0208
JRAlreadySigSetUp	513	0201
JRAlreadyTerminated	488	01E8
JRAnr	1123	0463
JRAsyncOpNotSupp	1045	0415
JRBadAddress	302	012E
JRBadAuditOption	336	0150
JRBadEntryCode	134	0086
JRBadID	978	03D2
JRBadIDType	880	0370
JRBadNETRCFile	2144	0860
JRBadNFSpathname	2142	085E
JRBadOptions	881	0371
JRBadTCPIPDATA	2143	085F
JRBadTCPXLBIN	2150	0866
JRBatSel	942	03AE
JRBindAliasUnsupported	2137	0859
JRBindBad24BitAddress	2136	0858
JRBindBadLIDXsegment	2129	0851
JRBindBadPOdata	2097	0831
JRBindBadRDT1outsideText	2132	0854
JRBindBadRDT2outsideText	2133	0855
JRBindBadRDT3BadFormat	2135	0857
JRBindBadRDT3outsideText	2134	0856
JRBindBadRDTFormat	2131	0853
JRBindBadRead	2053	0805
JRBindBadSegmentID	2130	0852
JRBindBadState	2049	0801
JRBindBadWrite	2054	0806

Table 10. OpenExtensions Reason Codes by Symbolic Name (continued)

Reason Code	Decimal	Hex
JRBindDuplicateModule	2112	0840
JRBindDuplicatePOKey	2098	0832
JRBindInvalidPOKey	2099	0833
JRBindMissingPOKey	2100	0834
JRBindNegativeLength	2080	0820
JRBindNoStorage	2064	0810
JRBindNoStoreNXext	2115	0843
JRBindNoStoreNXstd	2114	0842
JRBindNotClosed	2052	0804
JRBindNotOpenedI	2050	0802
JRBindNotOpenedO	2051	0803
JRBindNXstdModule	2113	0841
JRBindPSGMUnsupported	2128	0850
JRBothMode	105	0069
JRBrlmAlreadyWaiting	450	01C2
JRBrlmBadFileType	432	01B0
JRBrlmBadL_Len	445	01BD
JRBrlmBadL_Type	435	01B3
JRBrlmBadL_Whence	437	01B5
JRBrlmDeadLockDetected	440	01B8
JRBrlmFileLockRecycling	431	01AF
JRBrlmInvalidRange	436	01B4
JRBrlmNoReadAccess	433	01B1
JRBrlmNotActive	430	01AE
JRBrlmNoWriteAccess	434	01B2
JRBrlmObjAndProcBroken	458	01CA
JRBrlmProcessBroken	453	01C5
JRBrlmPromotePending	451	01C3
JRBrlmRangeNotAvailable	439	01B7
JRBrlmSignalPosted	441	01B9
JRBrlmUnlockWhileWait	457	01C9
JRBRMCancel	2107	083B
JRBrokenBrlmRecycling	490	01EA
JRBuffLenInvalid	277	0115

Table 10. OpenExtensions Reason Codes by Symbolic Name (continued)

Reason Code	Decimal	Hex
JRBuffTooSmall	107	006B
JRBytes2RWZero	138	008A
JRCallerIsPgLeader	508	01FC
JRCancel	1026	0402
JRChaudtoPipe	392	0188
JRChdNoEnt	80	0050
JRChdNotDir	79	004F
JRChmodFiletype	2051	0803
JRChowntoPipe	391	0187
JRClNeedClose	66	0042
JRClose	69	0045
JRCMSLoadFailure	945	03B1
JRCompNameTooLong	62	003E
JRCompNotDir	60	003C
JRConnectAuthFailure	2117	0845
JRConnectSSIFailure	2158	086E
JRCPInternalError	2006	07D6
JRCPNotAuthorized	2001	07D1
JRCPNotAvail	2002	07D2
JRCPNotFound	2000	07D0
JRCPUserNotFound	2007	07D7
JRCreate	76	004C
JRCtyAlreadyActive	2034	07F2
JRCtyBadQueSel	2026	07EA
JRCtyBgCall	2025	07E9
JRCtyConnectionInop	2018	07E2
JRCtyDeviceError	2033	07F1
JRCtyDiffSession	2021	07E5
JRCtyInputStopped	2029	07ED
JRCtyInvalidAction	2019	07E3
JRCtyInvalidPgid	2022	07E6
JRCtyNoCntlTerm	2020	07E4
JRCtyNoData	2032	07F0
JRCtyNotInSession	2023	07E7

Table 10. OpenExtensions Reason Codes by Symbolic Name (continued)

Reason Code	Decimal	Hex
JRCtyNotPGLeader	2024	07E8
JRCtyOrphanedRead	2031	07EF
JRCtyOrphanedWrite	2027	07EB
JRCtyOutputStopped	2030	07EE
JRCtySIGTTINBlocked	2028	07EC
JRDiffFileSets	123	007B
JRDirNotFound	61	003D
JRDirWriteRequest	100	0064
JRDomainNotSupported	552	0228
JRDotOrDotDot	130	0082
JRDuplicateCancel	1027	0403
JRDuplicateMEL	2081	0821
JRDup2Error	530	0212
JRECBerror	664	0298
JREcbError2	1500	05DC
JREcbWaitBitOn	669	029D
JREndingSlashExtlink	2100	0834
JREndingSlashMknod	264	0108
JREndingSlashOCreat	265	0109
JREndingSlashSymlink	515	0203
JRExecFileTooBig	337	0151
JRExecNmLenZero	271	010F
JRExecNotRegFile	287	011F
JRExtFileAlreadyExists	2096	0830
JRExtlink	2097	0831
JRFdAllocErr	136	0088
JRFdTooBig	325	0145
JRFd2TooSmall	461	01CD
JRFileDesNotInUse	55	0037
JRFileExistsExclFlagSet	99	0063
JRFileIsBlocked	312	0138
JRFileNotExtLink	2098	0832
JRFileNotOpen	284	011C
JRFileNotSymLink	261	0105

Table 10. OpenExtensions Reason Codes by Symbolic Name (continued)

Reason Code	Decimal	Hex
JRFileNotThere	108	006C
JRFilePoolSever	2114	0842
JRFilePoolUnknown	2113	0841
JRFileSpaceFull	2105	0839
JRFileSpaceUnknown	2082	0822
JRFilesysNotThere	46	002E
JRFsFailStorage	274	0112
JRFsNotRegFile	2078	081E
JRFSNotStart	90	005A
JRFsync	257	0101
JRFuncNotSupported	359	0167
JRGetAttr	110	006E
JRGetFirst	479	01DF
JRGetFQName	2080	0820
JRGetHostNameError	2139	085B
JRGIDLimitExceeded	2152	0868
JRHeavyWeight	478	01DE
JRIdentifyErr	2008	07D8
JRInactive	290	0122
JRIncorrectSocketType	617	0269
JRInternalError	132	0084
JRInvalidAmode	330	014A
JRInvalidAttr	2052	0804
JRInvalidAuthStruc	2071	0817
JRInvalidCjar	2053	0805
JRInvalidExtLinkLen	2099	0833
JRInvalidFileType	2055	0807
JRInvalidForSymlink	2056	0808
JRInvalidIName	2058	080A
JRInvalidInputBuf	2072	0818
JRInvalidMajorNumber	291	0123
JRInvalidMtab	2057	0809
JRInvalidNfds	954	03BA
JRInvalidOutputBuf	2073	0819

Table 10. OpenExtensions Reason Codes by Symbolic Name (continued)

Reason Code	Decimal	Hex
JRInvalidParms	169	00A9
JRInvalidPid	314	013A
JRInvalidResource	829	033D
JRInvalidRoutine	609	0261
JRInvalidSigAct	142	008E
JRInvalidSigHow	143	008F
JRInvalidSignal	141	008D
JRInvalidSymLinkComp	283	011B
JRInvalidSymLinkLen	282	011A
JRInvalidToken	2059	080B
JRInvalidUIO	2060	080C
JRInvalidVnode	168	00A8
JRInvCWD	2102	0836
JRInvDeviceId	2035	07F3
JRInvFilePoolID	2011	07DB
JRInvFileSpaceID	2012	07DC
JRInvOpenFlags	63	003F
JRInvOption	495	01EF
JRInvRoot	2103	0837
JRInvTermStat	319	013F
JRInvUserOp	642	0282
JRIOBufLengthInvalid	329	0149
JRIoctl	313	0139
JRIpcBadFlags	777	0309
JRIpcBadID	770	0302
JRIpcDenied	771	0303
JRIpcExists	772	0304
JRIpcMaxIDs	773	0305
JRIpcNoExist	774	0306
JRIpcRemoved	792	0318
JRIpcRetry	775	0307
JRIpcSignaled	776	0308
JRIsFSRoot	127	007F
JRIsMounted	91	005B

Table 10. OpenExtensions Reason Codes by Symbolic Name (continued)

Reason Code	Decimal	Hex
JRJoinToSelf	483	01E3
JRLastThread	551	0227
JRLightWeightThid	474	01DA
JRLink	270	010E
JRLinkNotFound	2036	07F4
JRListenAlreadyDone	653	028D
JRListenNotDone	649	0289
JRListLenBad	641	0281
JRListTooShort	612	0264
JRLnkAcrossFilesets	268	010C
JRLnkDir	86	0056
JRLnkNewPathExists	267	010B
JRLnkNoEnt	266	010A
JRLnkROFileset	269	010D
JRLockErr	175	00AF
JRLockRetryLim	2061	080D
JRLookup	71	0047
JRLskOffsetIsInvalid	88	0058
JRLskOnPipe	87	0057
JRLskWhenceIsInvalid	89	0059
JRMaxconnExceeded	2112	0840
JRMaxProc	40	0028
JRMkDir	68	0044
JRMkDirExist	56	0038
JRMkDirROnly	85	0055
JRMknodInvalidType	263	0107
JRMntNoPassword	2146	0862
JRMount	2083	0823
JRMountNotFQName	2083	0823
JRMountPt	161	00A1
JRMSOutOfRange	613	0265
JRMsq2Big	781	030D
JRMsqBadSize	779	030B
JRMsqBadType	778	030A

Table 10. OpenExtensions Reason Codes by Symbolic Name (continued)

Reason Code	Decimal	Hex
JRMsqNoMsg	780	030C
JRMsqQBytes	793	0319
JRMsqQueueFullBytes	797	031D
JRMsqQueueFullMessages	796	031C
JRMustBeSocket	547	0223
JRNanoSecondsTooBig	528	0210
JRNegativeValueInvalid	48	0030
JRNegFileDes	54	0036
JRNeitherMode	106	006A
JRNETRCFileNotFound	2145	0861
JRNetwork	553	0229
JRNewIsDir	125	007D
JRNewNotDir	124	007C
JRNFSCBranchFail	2118	0846
JRNFSCDecodeFail	2149	0865
JRNFSCInitFail	2119	0847
JRNFSCInitFail1	2120	0848
JRNFSCInitFail2	2121	0849
JRNFSCInitFail3	2122	084A
JRNFSCInitFail4	2123	084B
JRNFSCInitFail5	2124	084C
JRNFSCInitFail6	2125	084D
JRNFSCInitFail7	2126	084E
JRNFSCInitFail8	2127	084F
JRNFSCInitFail9	2133	0855
JRNFSCInitFail10	2134	0856
JRNFSCMntTCPIPDATA	2131	0853
JRNFSCMntTCXLBIN	2132	0854
JRNFSCNoPermMount	2155	086B
JRNFSCReqFail	2128	0850
JRNFSCReqFail1	2129	0851
JRNFSCReqFail2	2130	0852
JRNFSCSocketFail	2154	086A
JRNFSC2ManyRestart	2136	0858

Table 10. OpenExtensions Reason Codes by Symbolic Name (continued)

Reason Code	Decimal	Hex
JRNFSError	2137	0859
JRNFSMountError	2156	086C
JRNFSNamNotAllowed	2138	085A
JRNFSNotAllowed	2151	0867
JRNFSVerNotSupp	2148	0864
JRNoCTTY	425	01A9
JRNoData	309	0135
JRNoDFSMS	2106	083A
JRNoEvents	517	0205
JRNoExtLink	2084	0824
JRNoFdsTooManyQIds	961	03C1
JRNoFileNoCreatFlag	98	0062
JRNoMoreIOCache	2116	0844
JRNoMoreMtabs	2014	07DE
JRNoMorePNEs	2101	0835
JRNoMoreVFSs	2013	07DD
JRNoMoreVnods	2015	07DF
JRNoPath	77	004D
JRNoReaders	297	0129
JRNoRecall	2077	081D
JRNoSocket	546	0222
JRNoStorage	184	00B8
JRNoSuchPid	500	01F4
JRNotBFS	2062	080E
JRNotDescendant	506	01FA
JRNotForDir	144	0090
JRNothingMounted	2079	081F
JRNotPermitted	276	0114
JRNotSetup	519	0207
JRNotSigSetUp	514	0202
JRNotSupportedForFileType	281	0119
JRNotSysRoot	59	003B
JRNullInPath	58	003A
JRObjectInUse	2063	080F

Table 10. OpenExtensions Reason Codes by Symbolic Name (continued)

Reason Code	Decimal	Hex
JROK	0	0000
JROldNoExist	126	007E
JROldPartOfNew	145	0091
JROpen	75	004B
JROpenDirNotFound	114	0072
JROpenFlagConflict	101	0065
JROpenforWriteNoReaders	296	0128
JROpenMax	326	0146
JROutOfLocks	160	00A0
JROutOfOfteCells	95	005F
JROutofSocketNodeCells	593	0251
JRParmTooLong	103	0067
JRPathconf	837	0345
JRPathNotDir	120	0078
JRPathTooLong	57	0039
JRPCNFSError	2141	085D
JRPfsCtl	2135	0857
JRPfsDead	67	0043
JRPfsSuspend	182	00B6
JRPgidDifferentSession	507	01FB
JRPidDifferentSession	503	01F7
JRPidEQSessLeader	501	01F5
JRPipeProcErr	2108	083C
JRPortMapperError	2157	086D
JRPrevSockError	897	0381
JRPtatDetachState	499	01F3
JRPtatEye	467	01D3
JRPtatLen	493	01ED
JRPtatSyncType	498	01F2
JRPtatSysLen	492	01EC
JRPtatSysOff	491	01EB
JRPtCancelError	465	01D1
JRPtCreateError	462	01CE
JRPtExitError	464	01D0

Table 10. OpenExtensions Reason Codes by Symbolic Name (continued)

Reason Code	Decimal	Hex
JRQEFLErr	2010	07DA
JRQuiesced	180	00B4
JRQuiesceInProcess	550	0226
JRQuiesceTypeInvalid	549	0225
JRQuiescing	399	018F
JRRdandWRTforPipe	293	0125
JRRddFileNotDir	112	0070
JRRdlBuffLenInvalid	510	01FE
JRRdnorWRTforPipe	307	0133
JRRdwr	70	0046
JRReadDir	109	006D
JRReadlink	262	0106
JRReadOnlyFileSetCreatReq	97	0061
JRReadOnlyFileSetMknodReq	118	0076
JRReadOnlyFileSetWriteReq	96	0060
JRReadOnlyFS	121	0079
JRRemove	104	0068
JRRemoveTopDir	2064	0810
JRRename	128	0080
JRRenameTopDir	2110	083E
JRRFileWrOnly	51	0033
JRRmDir	119	0077
JRRoot	164	00A4
JRRootNode	140	008C
JRRoutineError	609	0261
JRRwdFileNotDir	139	008B
JRSecOutOfRange	614	0266
JRSema4BadAdj	782	030E
JRSema4BadNOps	783	030F
JRSema4BadNSems	784	0310
JRSema4BadSemN	786	0312
JRSema4BadValue	787	0313
JRSema4BigNSems	788	0314
JRSema4ZeroNSems	789	0315

Table 10. OpenExtensions Reason Codes by Symbolic Name (continued)

Reason Code	Decimal	Hex
JRSemStorageLimit	799	031F
JRSerStorageObtainErr	2111	083F
JRSetAttr	258	0102
JRSetpgidAfterSpawn	504	01F8
JRShmBadSize	790	0316
JRShmMaxAttach	791	0317
JRSigDuringWait	304	0130
JRSignalReceived	342	0156
JRSignalsNotBlocked	324	0144
JRSocketCallParmError	606	025E
JRSocketNamed	596	0254
JRSocketNotCon	626	0272
JRSocketTypeNotSupported	668	029C
JRSockNoName	632	0278
JRSockShutDown	636	027C
JRSoftLinkError	2065	0811
JRSpFileExists	117	0075
JRSrx	1124	0464
JRStackReadErr	2009	07D9
JRStatVFS	2147	0863
JRStorageGroupFull	2104	0838
JRStorageObtainErr	2066	0812
JRStorageReleaseErr	2067	0813
JRSvrMaxconnExceeded	2115	0843
JRSymFileAlreadyExists	259	0103
JRSymlink	260	0104
JRSyscallAbend	301	012D
JRTargetNotDir	113	0071
JRThreadNotFound	476	01DC
JRThreadTerm	472	01D8
JRTimeOut	529	0211
JRTooMany	502	01F6
JRTooManyFds	926	039E
JRTooManySymlinks	285	011D

Table 10. OpenExtensions Reason Codes by Symbolic Name (continued)

Reason Code	Decimal	Hex
JRTransportError	1503	05DF
JRTrMountedRO	157	009D
JRTrNegOffset	159	009F
JRTrNotRegFile	65	0041
JRTrOpenedRO	156	009C
JRTrunc	256	0100
JRUDCError1	2074	081A
JRUDCError2	2075	081B
JRUDCError3	2076	081C
JRUmount	162	00A2
JRUndefEvents	516	0204
JRUnexpectedErr	37	0025
JRUnexpectedError	177	00B1
JRUnlDir	94	005E
JrUnlMountRO	50	0032
JRUnlNoEnt	93	005D
JRUserNotAuthorized	310	0136
JRVnodGet	73	0049
JRWaitForever	953	03B9
JRWFileRdOnly	52	0034
JRWriteBeyondLimit	870	0366
JRWrongSsave	394	018A

Appendix C. System Control Offsets to Callable Services

An alternative to loading or link-editing the service stub is to include in the code the system control offset to the callable service.

When using the offsets, set the registers up as follows:

Register 0
To contain the service offset. For access (BPX1ACC), use 52 (decimal) for the offset. Another register can be used instead of register 0.

Register 1
To contain the address of your parameter list. Set bit 0 of the last address in the list on.

Register 14
To contain the return address in the invoking module.

Register 15
To contain the address of the callable service code.

The following is an example of code specifying the offset. Replace *offset* with the appropriate value from the following offset table:

```
LA    0,offset        Offset value
L     15,16           CVT - common vector table
L     15,544(,15)     CSRTABLE
L     15,24(,15)      OpenExtensions CSR slot
ALR   15,0            Add offset to base
L     15,0(,15)       Address of the service
BALR  14,15           Branch and link
```

Service	Offset	Function
BPX1ACC	52	access
BPX1ALR	224	alarm
BPX1CCA	480	cond_cancel
BPX1CHA	84	chaudit
BPX1CHD	56	chdir
BPX1CHM	60	chmod
BPX1CHO	64	chown
BPX1CLD	68	closedir
BPX1CLO	72	close
BPX1CPO	484	cond_post
BPX1CSE	488	cond_setup
BPX1CTE	237	create_thread_environment
BPX1CTW	492	cond_timed_wait
BPX1CWA	496	cond_wait
BPX1DEL	888	DLL_delete
BPX1EXC	228	exec
BPX1EXI	232	_exit
BPX1FCA	140	fchaudit
BPX1FCM	88	fchmod
BPX1FCO	92	fchown
BPX1FCT	96	fcntl
BPX1FPC	100	fpathconf
BPX1FRK	240	fork
BPX1FST	104	fstat
BPX1FSY	108	fsync
BPX1FTR	112	ftruncate
BPX1FTV	848	fstatvfs
BPX1GCW	116	getcwd
BPX1GEG	244	getegid
BPX1GET	736	w_getipc
BPX1GEU	248	geteuid
BPX1GGI	252	getgrgid
BPX1GGN	256	getgrnam
BPX1GGR	260	getgroups
BPX1GID	264	getgid
BPX1GLG	268	getlogin
BPX1GPG	272	getpgrp

© Copyright IBM Corp. 1993, 2022

```
BPX1GPI  276      getpid
BPX1GPN  280      getpwnam
BPX1GPP  284      getppid
BPX1GPS  428      w_getpsent
BPX1GPU  288      getpwuid
BPX1GUG  292      getgroupsbyname
BPX1GUI  296      getuid
BPX1ITY   12      isatty
BPX1KIL  308      kill
BPX1LNK  124      link
BPX1LOD  880      DLL_load
BPX1LSK  128      lseek
BPX1LST  132      lstat
BPX1MAT  720      shmat
BPX1MCT  724      shmctl
BPX1MDT  728      shmdt
BPX1MGT  732      shmget
BPX1MKD  136      mkdir
BPX1MKN  144      mknod
BPX1TFW   28      tcflow
BPX1MNT  148      mount
BPX1MPC  408      cmsprocclp
BPX1MSD  336      cmsunsigsetup
BPX1MSS  312      cmssigsetup
BPX1OPD  152      opendir
BPX1OPN  156      open
BPX1PAS  316      pause
BPX1PCF  160      pathconf
BPX1PIP  164      pipe
BPX1PSI  460      pthread_setintr
BPX1PST  472      pthread_setintrtype
BPX1PTB  448      pthread_cancel
BPX1PTC  432      pthread_create
BPX1PTD  444      pthread_detach
BPX1PTI  476      pthread_testintr
BPX1PTJ  440      pthread_join
BPX1PTK  464      pthread_kill
BPX1PTQ  412      quiesce_threads
BPX1PTS  452      pthread_self
BPX1PTX  436      pthread_exit_and_get
BPX1QCT  692      msgctl
BPX1QGT  696      msgget
BPX1QRC  700      msgrcv
BPX1QSN  704      msgsnd
BPX1RDD  168      readdir
BPX1RDL  172      readlink
BPX1RED  176      read
BPX1REN  180      rename
BPX1RMD  188      rmdir
BPX1RWD  184      rewinddir
BPX1SCT  708      semctl
BPX1SGT  712      semget
BPX1SOP  716      semop
BPX1SEG  424      setegid
BPX1SEU  420      seteuid
BPX1SGI  328      setgid
BPX1SIA  324      sigaction
BPX1SIP  340      sigpending
BPX1SLP  344      sleep
BPX1SPB  416      queue_interrupt
BPX1SPG  348      setpgid
BPX1SPM  352      sigprocmask
BPX1SPN  760      spawn
BPX1SSI  356      setsid
BPX1SSU  360      sigsuspend
BPX1STA  192      stat
BPX1STF   80      w_statfs
BPX1STV  844      statvfs
BPX1SUI  364      setuid
BPX1SWT  468      sigwait
BPX1SYC  368      sysconf
BPX1SYM  196      symlink
BPX1TDR   24      tcdrain
BPX1TFH   20      tcflush
BPX1TGA   32      tcgetattr
BPX1TGP   36      tcgetpgrp
BPX1TIM  372      times
BPX1TSA   40      tcsetattr
BPX1TSB   44      tcsendbreak
BPX1TSP   48      tcsetpgrp
BPX1TYN   16      ttyname
BPX1UMK  204      umask
```

```
BPX1UMT  208      umount
BPX1UNA  376      uname
BPX1UNL  212      unlink
BPX1UTI  216      utime
BPX1WAT  380      wait
BPX1WRT  220      write
BPX1WTE  840      'wait-extension'
```

Appendix D. Reentrant and Nonreentrant Linkage Examples

This appendix shows examples of reentrant and nonreentrant linkage.

Reentrant Entry Linkage

This entry linkage is reentrant and saves the caller's registers, allocates a save area and dynamic storage, and establishes program and dynamic storage base registers. This entry linkage is paired with the return linkage that is located at the end of the executable program. See "Reentrant Return Linkage" on page 551.

```
         TITLE 'Alphabetical Invocation of OpenExtensions Callable Services'
BOOKSAM1 CSECT ,                    Reentrant entry linkage
BOOKSAM1 AMODE 31
BOOKSAM1 RMODE ANY
         USING *,R15                Program addressability
@ENTRY0  B     @ENTRY1              Branch around program header
         DROP  R15                  R15 not needed for addressability
         DC    C'BOOKSAM1 - Reentrant callable service examples'
         DS    0H                   Ensure half word boundary
@ENTRY1  STM   R14,R12,12(R13)      Save caller's registers
         LR    R2,R13               Hold address of caller's area
         LR    R3,R1                Hold parameter register
         LR    R12,R15              R12 program base register
         LA    R11,2048(,R12)       Second program base register
         LA    R11,2048(,R11)       Second program base register
         USING @ENTRY0,R12,R11      Program addressability
         L     R0,@SIZEDAT          Size this program's getmain area
         GETMAIN RU,LV=(0)          Getmain storage
         LR    R13,R1               R13 -> this program's save area
         LA    R10,2048(,R13)       Second getmain base register
         LA    R10,2048(,R10)       Second getmain base register
         USING @STORE,R13,R10       Getmain addressability
         ST    R2,@BACK             Save caller's save area pointer
         ST    R13,8(,R2)           Give caller our save area
         LR    R1,R3                Restore parameter register
@ENTRY2  EQU   *  *  *  *  *  *  *   End of the entry linkage code
         SPACE ,
PSEUDO   EQU   *                    Dummy label used throughout
```

Reentrant Return Linkage

```
         XR    R15,R15              Zero return code
         L     R0,@SIZEDAT          Size this program's getmain area
         LR    R1,R13               R1 -> this program's getmain area
         L     R13,@BACK            R2 -> caller's save area
         DROP  R13
         FREEMAIN RU,LV=(0),A=(1)
         L     R14,12(,R13)         Restore caller's R14
         LM    R0,R12,20(R13)       Restore caller's R0-R12
         BSM   0,R14                Branch back to caller
         SPACE ,  * * * * * * * * * *.* Program constants * * * * * * *
@SIZEDAT DC    A(@ENDSTOR-@STORE)   Size of this getmain storage
MNTEL    DC    A(MNTE#LENGTH+MNTEH#LENGTH)
*                                   Length of MNTEH and 1 MNTE area
PGPSL    DC    A(PGPS#LENGTH)       Length of PGPS structure
RMONL    DC    A(RMON#LENGTH)       Length of RMON structure
SSTFL    DC    A(SSTF#LENGTH)       Length of SSTF structure
STATL    DC    A(STAT#LENGTH)       Length of STAT structure
UTSNL    DC    A(UTSN#LENGTH)       Length of UTSN structure
         SPACE ,
PRIMARYALET  DC   A(0)              Primary ALET
* * * * * * * * * * * * * * * * * * * *.* Structures requiring a USING  *
         BPXYDIRE DSECT=YES         Dictionary for readdir
         BPXYGIDN DSECT=YES         Group names
         BPXYGIDS DSECT=YES         Group IDs and member names
         BPXYOSMF DSECT=YES         Job step accounting for BPXESMF
```

© Copyright IBM Corp. 1993, 2022

```
         BPXYPPSD DSECT=YES          Signal data area (R1 in SIR)
* * * * * * * * * * * * * * * * * * *.* OpenExtensions EQUates * * * * * * *
* With EQUate only macros, DSECT= is allowed but is ignored
         BPXYCONS ,                  OpenExtensions constants
         BPXYCW   ,                  Serialization constants
         BPXYERNO LIST=NO            Errno, Errnojr constants
         BPXYFTYP ,                  File type constants
         BPXYPCF  ,                  Command, pathconf constants
         BPXYSEEK ,                  lseek constants
         BPXYSIGH ,                  Signal constants
```

```
* * * * * * * * * * * * * * * * * * * *.* Standard linkage save area  * *
@STORE   DSECT ,
@SAVE00  DS    0D                    Standard 72-byte save area
         DS    A
@BACK    DS    A                     Back to caller's save area
@FORWARD DS    A                     Forwards to callee's save area
         DS    15A                   Regs 14,15,0-12
```

```
         SPACE 2  * * * * * * * * * *.* Getmain for mappings * * * * *
         BPXYACC  DSECT=NO           Access intent flags
         BPXYAUDT DSECT=NO           Audit flage values for chaudit
BRLKA    DS    A                     ->BPXYBRLK
         BPXYBRLK DSECT=NO           Byte range locking for fcntl
         BPXYFCTL DSECT=NO           Flags and commands for fcntl
         BPXYMODE DSECT=NO           Mode constants
         BPXYMTM  DSECT=NO           Mount/unmount modes
         BPXYOPNF DSECT=NO           File open constants
PGPSA    DS    A                     ->BPXYPGPS
         BPXYPGPS DSECT=NO,          Process slot data, w_getpsent     +
               VARLEN=(,0,0)         Contty=Default, Path=0, Cmd=0
PTATA    DS    A                     ->BPXYPTAT
         BPXYPTAT DSECT=NO,VARLEN=512  Pthreat attributes
PTXLA    DS    A                     ->BPXYPTXL
         BPXYPTXL DSECT=NO           Pthread attribute area
         BPXYRMON DSECT=NO           Resource monitor
         BPXYSTAT DSECT=NO           Get file the status for stat
         BPXYTIMS DSECT=NO           times callable service structure
TIOS     BPXYTIOS DSECT=NO           Termios structure
UTSNA    DS    A                     ->BPXYUTSN
         BPXYUTSN DSECT=NO           uname structure
WASTA    DS    A                     ->BPXYWAST
         BPXYWAST DSECT=NO           Status word for wait
```

```
* * * * * * * * * * * * * * * * * * *.* Program getmain variables * * *
             DS    0D
ARGCNT       DS    F                 Argument count
ARGLLST      DS    3A                Argument lengths list
ARGSLST      DS    3A                Arguments list
BUFA         DS    F                 ->buffer
BUFFERA      DS    CL1024            Utility buffer A, length 1024
BUFLENA      DS    F                 Number of bytes used in buffer A
BUFFERB      DS    CL1024            Utility buffer B, length 1024
BUFLENB      DS    F                 Number of bytes used in buffer B
BUFW         DS    F                 Number of words used in BUF
COMMAND      DS    F                 User defined command
DIRECTDES    DS    F                 Directory descriptor
ENVCNT       DS    F                 Number of environment variables
ENVLENS      DS    F                 Length of environment variables
ENVPARMS     DS    F                 Environment variables
EVENTLIST    DS    A                 Event list for thread posting
EXITRTNA     DS    A                 Exit routine address
EXITPLA      DS    A                 Exit Parm list address
FILEDESC     DS    F                 File descriptor
FILEDES2     DS    F                 File descriptor
FSNAME       DS    CL44              File system name
FSTYPE       DS    CL8               File system type
GRNAMELN     DS    F                 Group name length
GROUP        DS    F                 Group
GROUPCNT     DS    F                 Group count
GROUPID      DS    F                 Group ID (PID of group leader)
GRPGMNAME    DS    CL8               Group program name
INTMASK      DS    XL8               Signal mask
INITRTNA     DS    A                 ->Initialization routine
INTRSTATE    DS    A                 Interrupt state
INTRTYPE     DS    A                 Interrupt type
NANOSECONDS  DS    F                 Count of nanoseconds
NCATCHER     DS    A                 New catcher
NEWFLAGS     DS    F                 New flags
```

```
NEWHANDL      DS    F               New Handler
NEWLEN        DS    XL8             Length file
NEWMASK       DS    XL8             New mask for signals
NEWMASKA      DS    A               ->New mask
NEWTIMES      DS    D               New access/modification time
OCATCHER      DS    A               Old catcher
OFFSET        DS    CL8             File offset
OLDHANDL      DS    F               Old handler
OLDFLAGS      DS    F               Old flags
OLDMASK       DS    CL8             Old signal mask
OLDMASKA      DS    A               ->Old mask
OPTIONS       DS    F               Options
PGMNAME       DS    CL8             Program name
PGMNAMEL      DS    F               Length PGMNAME
PLIST         DS    13A             Max number of parms
PROCID        DS    F               Process ID
PROCTOK       DS    F               Relative process number
READFD        DS    F               File descriptor - input file
REFPT         DS    F               File reference point
RETCODE       DS    F               Return code (ERRNO)
RETVAL        DS    F               Return value (0, -1 or other)
RSNCODE       DS    F               Reason code (ERRNOJR)
SECONDS       DS    F               Time in seconds
SIGNAL        DS    A               Signal
SIGNALREG     DS    A               Signal setup, user data
SIGNALOPTIONS DS    A               Signal options
SIGRET        DS    CL8             Signal return mask
SIRTNA        DS    A               Signal interrupt routine
STATFLD       DS    A               Status field
STATUS        DS    F               Status
STATUSA       DS    A               ->STATUS
TERMMASK      DS    XL8             Signal termination mask
THID          DS    XL8             Thread ID
USERID        DS    F               User ID
USERNAME      DS    CL8             User name
USERNLEN      DS    F               Length USERNAME
USERWORD      DS    F               User data
WAITMASK      DS    F               Mast for signal waits
WRITEFD       DS    F               File descriptor - output file
              SPACE ,
@ENDSTOR      EQU   *               End of getmain storage
         SPACE 3  * * * * * * * * * *.* Register equates  * * * * * * *
              SPACE ,
R0            EQU   0
R1            EQU   1               Parameter list pointer
R2            EQU   2
R3            EQU   3
R4            EQU   4
R5            EQU   5
R6            EQU   6
R7            EQU   7
R8            EQU   8
R9            EQU   9
R10           EQU   10              Second getmain storage register
R11           EQU   11              Second program base register
R12           EQU   12              Program base register
R13           EQU   13              Savearea & getmain storage base
R14           EQU   14              Return address
R15           EQU   15              Branch location
              END
```

Nonreentrant Entry Linkage

This example shows the function for the w_getpsent (BPX1GPS) service in a nonreentrant program. For a reentrant example of this service, see “w_getpsent (BPX1GPS) -- Get Process Data” on page 394.

```
BOOKSAM3 CSECT ,                     Nonreentrant linkage
BOOKSAM3 AMODE 31
BOOKSAM3 RMODE ANY
         USING *,R15                 Program addressability
@BEGIN0  B     @BEGIN1               Branch around program header
         DC    C'BOOKSAM3 - nonreentrant w-getpsent invoker'
         DS    0H
@BEGIN1  STM   R14,12,12(R13)        Save caller's registers
         ST    R13,@BACK             Save ->Caller's save area
         LA    R13,@SAVE00           R13 program and save area base
         DROP  R15
         USING @SAVE00,R13           Program addressability
```

```
         B     @BEGIN2
@SAVE00  DS    0D                    Standard save area - 72 Bytes
         DS    A
@BACK    DS    A                     Backwards save area pointer
@FORWARD DS    A                     Forwards save area pointer
         DS    15A                   Regs 14,15,0-12
RETURN   XR    R15,R15               Zero return code
RETURNRC L     R13,@BACK             Restore caller's R13
         L     R14,12(,R13)          Restore caller's R14
         LM    R0,R12,20(R13)         Restore caller's R0-R12
         BSM   0,R14                 Branch back to caller
R0       EQU   0
R1       EQU   1                     Parameter list pointer
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13                    Program and save area base
R14      EQU   14                    Return address
R15      EQU   15                    Branch location
@BEGIN2  EQU   *  *  *  *  *  *  *   End of the entry linkage code
```

Fields PGPSCONTTYBLEN, PGPSCONTTYPTR, PGPSPATHBLEN, PGPSPATHPTR, PGPSCMDBLEN and PGPSCMDPTR are initialized by the expansion of the BPXYPGPS macro when expanded in a CSECT. Likewise fields PGPSA and PGPSA can also be initialized before the program runs. Contrast this with the reentrant example where these fields must be set by the program while it runs. These fields could also be initialized during execution in this, the nonreentrant example.

```
GETPS    L     R15,=V(BPX1GPS)       Address of BPX1GPS load module
         CALL  (15),                 Get process data                  +
               (PROCTOKEN,           Relative process token            +
               PGPSL,                Length of buffer                  +
               PGPSA,                Buffer, mapped by BPXYPGPS        +
               RETVAL,               Return value (next, eof or error) +
               RETCODE,              Return code                       +
               RSNCODE),             Reason code                       +
               VL                    ----------------------------------
         SPACE ,  *  *  *  *  *  *   Test for end of file
         ICM   R15,B'1111',RETVAL    Load return value, set CCode
         BZ    RETURN                0 is end of file
         BL    RETURNRC              -1 is error
         ST    R15,PROCTOKEN         Store the next process token
         SPACE ,  *  *  *  *  *  *   Initialize WTO area & message
         MVI   XPID,C' '             Blank out variable portion message
         MVC   XPID+1(WTO#BLANK-1),XPID
         SPACE ,  *  *  *  *  *  *   Process ID to printable hex
         L     R8,PGPSPID            R8 = process ID
         LA    R9,XPID               To be placed at message start
         LA    R15,8                 8 nibbles to convert (4 bytes)
         LA    R10,9                 For 0-9 / A-F compare
NIBBLE   LR    R11,R8                Target bits in 0-3     XYYYYYYZ
         SRL   R11,28                Bits 0-3 to 28-31      0000000X
         SLL   R8,4                  Drop bits 0-3 off end  YYYYYYZ0
         CLR   R11,R10               Are 4 bits 0-9 or A-F
         BC    B'0010',AF            Branch if A-F
         LA    R11,57(,R11)          Add for 0-9 (57+183=240 or F0)
AF       LA    R11,183(,R11)         Add for 0-F (183+10=193 or C1)
         STC   R11,0(,R9)            Store to results location
         LA    R9,1(,R9)             Increment R9 to next location
         BCT   R15,NIBBLE            Decrement half byte counter, loop
         SPACE ,  *  *  *  *  *  *   Test status bits
* Go after the state of the process
         MVI   THREAD,C'1'                Assume single task thread
         TM    PGPSSTATUS1,PGPSMULTHREAD  if multithread process
         BZ    NOTMULT
         MVI   THREAD,C'M'
NOTMULT  TM    PGPSSTATUS1,PGPSPTHREAD    if pthread_create tasks
         BZ    NOTIPT
         MVI   THREAD,C'H'
NOTIPT   MVC   STATE,PGPSSTATUS3          Z, W, X, S, C, F, K, R
         TM    PGPSSTATUS0,PGPSSWAP       if swapped out
         BZ    NOTSWAP
```

```
         MVC   SWAPA,=CL4'SWAP'
NOTSWAP  TM    PGPSSTATUS1,PGPSSTOPPED    if stopped
         BZ    NOTSTOP
         MVC   STOPA,=CL4'STOP'
NOTSTOP  EQU   *
         SPACE ,  *  *  *  *  *  *   Display message to operator
         LA    R2,WTOAREA            R2->WTO message area
         WTO   TEXT=(R2)             Write to Operator
         SPACE ,  *  *  *  *  *  *   Loop back
         B     GETPS                 for the next Process data
         SPACE ,
WTOAREA  DS    0F                    WTO message
         DC    AL2(WTO#LENGTH)         Length of area
         DC    CL4'PID='               Process ID =
XPID     DS    CL8                   Hex of process ID
         DS    CL1
THREAD   DS    CL1                   1, M or H
         DS    CL1
STATE    DS    CL1                   Z, W, X, S, C, F, K, R
         DS    CL1
SWAPA    DS    CL4                   SWAP or blank
         DS    CL1
STOPA    DS    CL4                   STOP or blank
         DS    CL1
TRACA    DS    CL4                   TRAC or blank
WTO#BLANK EQU  *-XPID                Length to blank
         DC    C'.'
WTO#LENGTH EQU *-WTOAREA             Length of WTO area
         SPACE ,
GPSENTRY DS    A                     Address of BPX1GPS
PROCTOKEN DC   A(0)                  Relative process token init to 0
RETVAL   DS    F                     Return value - next PROCTOKEN
RETCODE  DS    F                     Return code
RSNCODE  DS    F                     Reason code
         SPACE ,
PGPSL    DC    A(PGPS#LENGTH)        Length of PGPS buffer
PGPSA    DC    A(PGPS)               ->Process data buffer
         BPXYPGPS DSECT=NO,          Place in current CSECT / DSECT    +
               VARLEN=(0,0,0)         ConTty, Path, Cmd not needed
         END
```

Appendix E. The Relationship of OpenExtensions Signals to Callable Services

Before reading this information, you should read the signal information in the POSIX .1 standard and *z/VM: OpenExtensions POSIX Conformance Document*. The signal information in this appendix is the information needed by compiler writers implementing POSIX in a high-level language.

Signals support the following callable services:

"alarm (BPX1ALR) — Set an Alarm" on page 18
"kill (BPX1KIL) — Send a Signal to a Process" on page 146
"cmsunsigsetup (BPX1MSD) — Detach the Signal Setup" on page 44
"cmssigsetup (BPX1MSS) — Set Up CMS Signals" on page 40
"pause (BPX1PAS) — Suspend a Process Pending a Signal" on page 197
"pthread_kill (BPX1PTK) — Send a Signal to a Thread" on page 214
"sigaction (BPX1SIA) — Examine or Change a Signal Action" on page 315
"sigpending (BPX1SIP) — Examine Pending Signals" on page 319
"sleep (BPX1SLP) — Suspend Execution of a Process for an Interval of Time" on page 328
"queue_interrupt (BPX1SPB) — Return the Last Interrupt Delivered" on page 223
"sigprocmask (BPX1SPM) — Examine or Change a Thread's Signal Mask" on page 321
"sigsuspend (BPX1SSU) — Change the Signal Mask and Suspend the Thread Until a Signal Is Delivered" on page 324
"sigwait (BPX1SWT) — Wait for a Signal" on page 326.

High-Level-Language Signal Interfaces

In addition to the signal interface callable services defined by POSIX, OpenExtensions provides the following signal interface services:

cmssigsetup service

Sets up and defines the *signal interface routine (SIR)*. The SIR is a routine provided by the high-level language. For information on how to write the SIR and the interface to it, see "3" on page 41.

cmsunsigsetup service

Detaches the interface to the SIR and returns the parameters set up in cmssigsetup. See "cmsunsigsetup (BPX1MSD) — Detach the Signal Setup" on page 44.

queue_interrupt service

Returns the last signal delivered. See "queue_interrupt (BPX1SPB) — Return the Last Interrupt Delivered" on page 223.

These interfaces allow a runtime library (RTL) for a high-level language to control the flow of signals. Each high-level language defines its own linkage interface between callable procedures; for example, the C language has a linkage stack and register interface between function procedures, which are unique to C.

Delivery of signals involves:

- Interrupting a currently running procedure
- Saving the status of the code that was interrupted
- Invoking a callable procedure known as the signal catcher, or signal handler.

How High-Level Languages Use Signals

Invoking a callable service involves setting up registers unique to the high-level language.

© Copyright IBM Corp. 1993, 2022

1. The RTL, using these callable services, sets up a SIR to receive control when a signal occurs.
2. The SIR procedure performs the necessary language linkages and POSIX functions to call the signal catcher procedure.
3. The signal catcher may return to the SIR. Information passed to the SIR procedure is mapped by the BPXYPPSD macro.
4. The SIR performs the necessary language and POSIX functions to return to the interrupted procedure after the signal catcher returns.
5. The CSRL16J system service is used to load all registers and the PSW condition code and to jump to the instruction that was interrupted by the signal.

Signal Setup When Linking to Callable Services

When a CMS thread invokes the first OpenExtensions call, the containing CMS process and session (i.e. the virtual configuration) is implicitly set up for OpenExtensions callable services. This setup operation may also be performed explicitly by invoking the BPX1CTE service. The setup operation causes a new POSIX process to be created and assigned a unique POSIX process ID. Each thread in a POSIX process is additionally assigned an 8-character thread ID. This thread ID is unique within the process, though threads in different processes could have the same thread ID.

The first OpenExtensions call in a POSIX(ON) program invokes the cmssigsetup service. This establishes the POSIX environment and sets up the thread for signals at its current CMS SVC level.

Figure 3 on page 558 shows the flows for the various signal functions when a synchronous signal **SIGPIPE** is generated with the kill service.

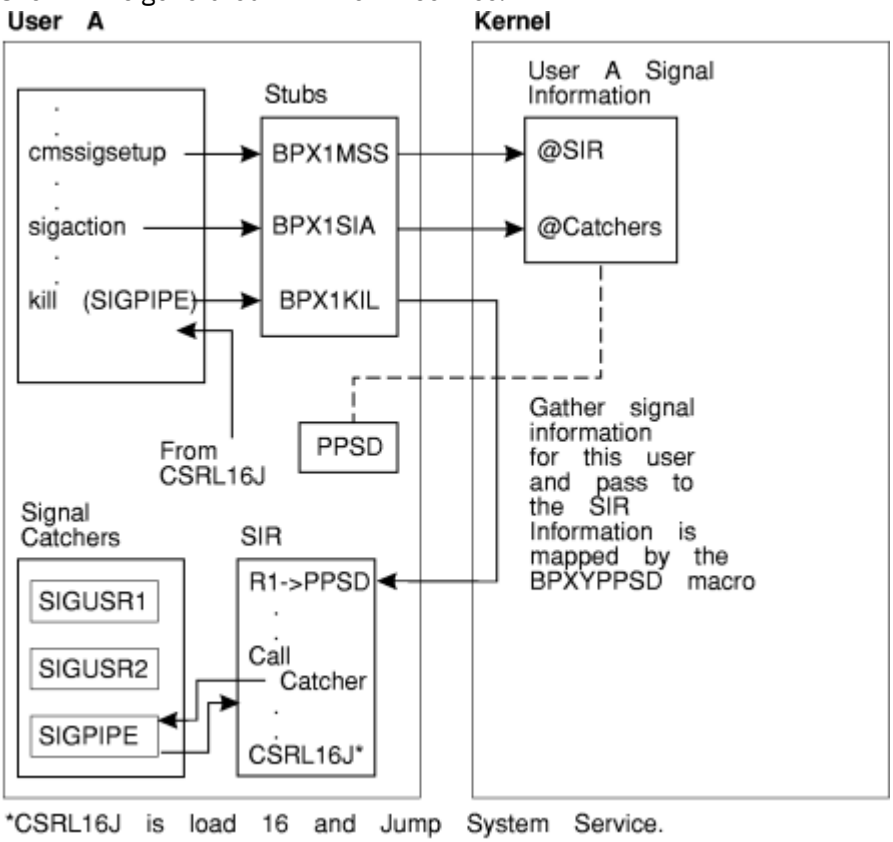


Figure 3. Program Flow of cmssigsetup and sigaction with Signal Interface Routine (SIR)

For more information on the set up and use of SIRs, see “cmssigsetup (BPX1MSS) — Set Up CMS Signals” on page 40. For more information on signal catchers, see “sigaction (BPX1SIA) — Examine or Change a Signal Action” on page 315.

VMERROR Event Handling and the SIGILL, SIGFPE, and SIGSEGV Signals

High-level languages generate the **SIGILL**, **SIGFPE**, and **SIGSEGV** signals. In OpenExtensions, the kill service is invoked to generate these signals. A VMERROR event handler may also use the kill service to generate **SIGILL**, **SIGFPE**, and **SIGSEGV**. High-level languages can define a VMERROR event handling routine to receive control after an incorrect hardware instruction, arithmetic operation, and memory reference.

Since OpenExtensions does not generate or process the signals **SIGILL**, **SIGFPE**, and **SIGSEGV**, it is the responsibility of the high-level language's RTL to define what happens when a signal catcher is defined for these signals and the signal catcher returns to the failing instruction.

When Signals Are Supported and Not Supported

All signal functions are supported when the thread is set up for signals, and the thread's current CMS SVC level is the same as when the thread was set up for signals. When this is not the case, some signal functions are not supported or they function differently. Table 11 on page 559 defines these signal functions.

The cmssigsetup columns in Table 11 on page 559 describe a thread that is set up with the cmssigsetup service. When a thread invokes the cmssigsetup service, the current SVC level is recorded for future signal delivery. When a thread has been set up for signals by cmssigsetup, signals are delivered to it only when the thread's current SVC level is the same as the SVC level at which it called cmssigsetup.

Table 11. Support of Signal Calls

Service	Thread cmssigsetup		Thread Not cmssigsetup	
	Current SVC Level Called cmssigsetup	**Current SVC Level Did Not Call cmssigsetup**	**Current SVC Level Called BPX1CTE**	**Current SVC Level Did Not Call BPX1CTE**
BPX1ALR	RV=Seconds	Abend	RV=Seconds	Abend
BPX1KIL	RV=0	RV=0	RV=0	RV=0
BPX1MSD	RV=0	RV=0	RV=-1	RV=-1
BPX1MSS	RV=-1	RV=-1	RV=0	RV=0
BPX1PAS	RV=0	RV=-1	RV=0	RV=0
BPX1SIA	RV=0	RV=-1	RV=-1	RV=-1
BPX1SIP	RV=0	RV=-1	RV=0	RV=0
BPX1SLP	RV=Seconds	RV=Abend	RV=Seconds	RV=Seconds
BPX1SPB	RV=0	N/A	N/A	N/A
BPX1SPM	RV=0	RV=-1	RV=0	RV=0
BPX1SSU	RV=0	RV=-1	RV=0	RV=0

Table 11. Support of Signal Calls (continued)

Service	Thread cmssigsetup		Thread Not cmssigsetup	
	Current SVC Level Called cmssigsetup	Current SVC Level Did Not Call cmssigsetup	Current SVC Level Called BPX1CTE	Current SVC Level Did Not Call BPX1CTE

Note:

SVC level
CMS SVC levels are created by the CMSCALL service and also by simulated MVS system services such as LINK.

RV
Return value returned in the service.

N/A
Not applicable

Delayed Signal Delivery

Asynchronous signals are generated from a process or thread different from the thread the signal is being delivered to. Delivery of asynchronous signals is not always possible immediately and may experience some delay. In particular, a signal may not be delivered to a thread which is executing in the CMS kernel or with PSW key 0; in that case, signal delivery is delayed until the thread exits from the kernel. Signals that must be delayed are delivered later, when signals are permitted and an opportunity for signal delivery arises.

Additionally, when a thread that is set up for signals by a cmssigsetup service issues a CMSCALL or other system service call (for example, LINK) that creates another CMS SVC level, delivery of signals to that thread is delayed until the thread returns to the registered SVC level and issues an OpenExtensions system call.

When Signals Cannot Be Delivered

Compilers and applications that enter states when signals cannot be delivered should invoke OpenExtensions callable services after returning to a state where signal delivery is possible. This action ensures prompt delivery of signals. For example, a program may invoke a CMSSTOR OBTAIN and getpid service. After returning from the getpid service, OpenExtensions delivers any asynchronous signals that were generated during the CMSSTOR OBTAIN.

When the SIR is unable to deliver a signal to a signal catcher routine for environmental reasons, the queue_interrupt service is invoked from a signal interface routine (SIR). The queue_interrupt service also delays signal delivery until the next OpenExtensions callable service. OpenExtensions callable services should be performed shortly after a queue_interrupt call to ensure prompt signal delivery.

Signals and Multiple Threads Created by ThreadCreate

The first POSIX thread in a process can be created either explicitly with the spawn (BPX1SPN), exec (BPX1EXC), or create_thread_environment (BPX1CTE) callable service, or implicitly by the first call to any other OpenExtensions callable service from any thread in that CMS session. Subsequent CMS threads can be created in the process with the ThreadCreate callable service.

The cmssigsetup and sigaction services allow only one thread in a process to set up a signal interface routine (SIR) and signal catchers. When a process contains two threads with signals unblocked, the signal is delivered to the thread that called cmssigsetup.

If signal action on delivery of a signal specifies termination, stop, or continue, the entire process is terminated, stopped, or continued. Delivery of a signal for default signal action occurs for any of the following conditions:

1. None of the threads is set up for signals by cmssigsetup and one or more threads do not have the signal blocked.
2. One of the threads is set up for signals by cmssigsetup and the signal is not blocked by the thread that called cmssigsetup.

Signals and Multiple Threads Created by pthread_create

The pthread_create service creates POSIX threads within the process. A thread created by pthread_create also inherits any signal setup information established by a prior cmssigsetup call in the creating thread. If the caller of pthread_create had previously called cmssigsetup successfully, the thread created is also set up for signals. The cmssigsetup and pthread_create services can be used to create multiple threads that are set up for signals in the same process.

When a signal is generated by a kill service request to a process that has some threads which are set up for signals and other threads which are not set up for signals, OpenExtensions signal processing must determine which thread has the most interest in the signal. The following is a list of signal interest rules for a signal generated by a kill call from most to least interested:

1. When threads are found in a sigwait for this signal, the signal is delivered to the first thread found in a sigwait.
2. When all threads are blocking this signal, the signal is left pending at the process level. The sigpending service moves blocked pending signals at the process level to the thread level.
3. When the default terminating signal action (not ignore and not catch) is to take place, that action is performed for all threads in the process.
4. When all of the following are true:
   - One or more threads are set up for signals.
   - All threads set up for signals have the signal blocked.
   - A thread not set up for signals has not blocked the signal.

   The signal is left pending on the first thread set up for signals, and remains pending on that thread until the thread unblocks the signal.
5. When one or more threads are set up for signals and at least one of the threads set up for signals has the signal unblocked, the signal is delivered to the first thread that is set up for signals that also has the signal unblocked.

Signal Defaults

This section contains information on the signals supported by OpenExtensions. These signals are mapped by the BPXYSIGH mapping macro; see “BPXYSIGH — Map Signal Constants” on page 462. The following table lists the signals supported by OpenExtensions and their default actions:

Constant	Value	Default Action	Description
SIGABRT#	3	1	Abnormal termination
SIGALRM#	14	1	Timeout
SIGFPE#	8	1	Erroneous arithmetic operation, such as division by zero or an operation resulting in overflow
SIGHUP#	1	1	Hangup detected on controlling terminal
SIGILL#	4	1	Detection of an incorrect hardware instruction
SIGINT#	2	1	Interactive attention
SIGKILL#	9	1	Termination (cannot be caught or ignored)
SIGPIPE#	13	1	Write on a pipe with no readers

Constant	Value	Default Action	Description
SIGQUIT#	24	1	Interactive termination
SIGSEGV#	11	1	Detection of an incorrect memory reference
SIGTERM#	15	1	Termination
SIGUSR1#	16	1	Reserved as application-defined signal 1
SIGUSR2#	17	1	Reserved as application-defined signal 2
SIGCHLD#	20	2	Child process terminated or stopped
SIGCONT#	19	4	Continue if stopped
SIGSTOP#	7	3	Stop (cannot be caught or ignored)
SIGTSTP#	25	3	Interactive stop
SIGTTIN#	21	3	Read from a controlling terminal attempted by a member of a background process group
SIGTTOU#	22	3	Write from a controlling terminal attempted by a member of a background process group
SIGNULL#	0	2	Null; no signal sent (cannot be caught or ignored)
SIGIO#	23	2	Completion of input or output
SIGABND#	18	1	Abend

The default actions are:

1. Abnormal termination.
2. Ignore the signal.
3. Stop the process.
4. Continue if it is currently stopped; otherwise, ignore the signal.

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

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

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

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

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

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 Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

© Copyright IBM Corp. 1993, 2022

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

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

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Programming Interface Information

This publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of z/VM.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., in the United States and/or other countries. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on IBM Copyright and trademark information (https://www.ibm.com/legal/copytrade).

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

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

Terms and Conditions for Product Documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal Use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see:

- The section entitled **IBM Websites** at IBM Privacy Statement (https://www.ibm.com/privacy)
- Cookies and Similar Technologies (https://www.ibm.com/privacy#Cookies_and_Similar_Technologies)

Bibliography

This topic lists the publications in the z/VM library. For abstracts of the z/VM publications, see *z/VM: General Information*.

Where to Get z/VM Information

The current z/VM product documentation is available in IBM Documentation - z/VM (https://www.ibm.com/docs/en/zvm).

z/VM Base Library

Overview

- *z/VM: License Information*, GI13-4377
- *z/VM: General Information*, GC24-6286

Installation, Migration, and Service

- *z/VM: Installation Guide*, GC24-6292
- *z/VM: Migration Guide*, GC24-6294
- *z/VM: Service Guide*, GC24-6325
- *z/VM: VMSES/E Introduction and Reference*, GC24-6336

Planning and Administration

- *z/VM: CMS File Pool Planning, Administration, and Operation*, SC24-6261
- *z/VM: CMS Planning and Administration*, SC24-6264
- *z/VM: Connectivity*, SC24-6267
- *z/VM: CP Planning and Administration*, SC24-6271
- *z/VM: Getting Started with Linux on IBM Z*, SC24-6287
- *z/VM: Group Control System*, SC24-6289
- *z/VM: I/O Configuration*, SC24-6291
- *z/VM: Running Guest Operating Systems*, SC24-6321
- *z/VM: Saved Segments Planning and Administration*, SC24-6322
- *z/VM: Secure Configuration Guide*, SC24-6323

Customization and Tuning

- *z/VM: CP Exit Customization*, SC24-6269
- *z/VM: Performance*, SC24-6301

Operation and Use

- *z/VM: CMS Commands and Utilities Reference*, SC24-6260
- *z/VM: CMS Primer*, SC24-6265
- *z/VM: CMS User's Guide*, SC24-6266
- *z/VM: CP Commands and Utilities Reference*, SC24-6268

© Copyright IBM Corp. 1993, 2022

- *z/VM: System Operation*, SC24-6326
- *z/VM: Virtual Machine Operation*, SC24-6334
- *z/VM: XEDIT Commands and Macros Reference*, SC24-6337
- *z/VM: XEDIT User's Guide*, SC24-6338

Application Programming

- *z/VM: CMS Application Development Guide*, SC24-6256
- *z/VM: CMS Application Development Guide for Assembler*, SC24-6257
- *z/VM: CMS Application Multitasking*, SC24-6258
- *z/VM: CMS Callable Services Reference*, SC24-6259
- *z/VM: CMS Macros and Functions Reference*, SC24-6262
- *z/VM: CMS Pipelines User's Guide and Reference*, SC24-6252
- *z/VM: CP Programming Services*, SC24-6272
- *z/VM: CPI Communications User's Guide*, SC24-6273
- *z/VM: ESA/XC Principles of Operation*, SC24-6285
- *z/VM: Language Environment User's Guide*, SC24-6293
- *z/VM: OpenExtensions Advanced Application Programming Tools*, SC24-6295
- *z/VM: OpenExtensions Callable Services Reference*, SC24-6296
- *z/VM: OpenExtensions Commands Reference*, SC24-6297
- *z/VM: OpenExtensions POSIX Conformance Document*, GC24-6298
- *z/VM: OpenExtensions User's Guide*, SC24-6299
- *z/VM: Program Management Binder for CMS*, SC24-6304
- *z/VM: Reusable Server Kernel Programmer's Guide and Reference*, SC24-6313
- *z/VM: REXX/VM Reference*, SC24-6314
- *z/VM: REXX/VM User's Guide*, SC24-6315
- *z/VM: Systems Management Application Programming*, SC24-6327
- *z/VM: z/Architecture Extended Configuration (z/XC) Principles of Operation*, SC27-4940

Diagnosis

- *z/VM: CMS and REXX/VM Messages and Codes*, GC24-6255
- *z/VM: CP Messages and Codes*, GC24-6270
- *z/VM: Diagnosis Guide*, GC24-6280
- *z/VM: Dump Viewing Facility*, GC24-6284
- *z/VM: Other Components Messages and Codes*, GC24-6300
- *z/VM: VM Dump Tool*, GC24-6335

z/VM Facilities and Features

Data Facility Storage Management Subsystem for z/VM

- *z/VM: DFSMS/VM Customization*, SC24-6274
- *z/VM: DFSMS/VM Diagnosis Guide*, GC24-6275
- *z/VM: DFSMS/VM Messages and Codes*, GC24-6276
- *z/VM: DFSMS/VM Planning Guide*, SC24-6277

- *z/VM: DFSMS/VM Removable Media Services*, SC24-6278
- *z/VM: DFSMS/VM Storage Administration*, SC24-6279

Directory Maintenance Facility for z/VM

- *z/VM: Directory Maintenance Facility Commands Reference*, SC24-6281
- *z/VM: Directory Maintenance Facility Messages*, GC24-6282
- *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*, SC24-6283

Open Systems Adapter

- Open Systems Adapter-Express Customer's Guide and Reference (https://www.ibm.com/support/pages/node/6019492), SA22-7935
- Open Systems Adapter-Express Integrated Console Controller User's Guide (https://www.ibm.com/support/pages/node/6019810), SC27-9003
- Open Systems Adapter-Express Integrated Console Controller 3215 Support (https://www.ibm.com/docs/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.ioa/ioa.htm), SA23-2247
- Open Systems Adapter/Support Facility on the Hardware Management Console (https://www.ibm.com/docs/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.ioa/ioa.htm), SC14-7580

Performance Toolkit for z/VM

- *z/VM: Performance Toolkit Guide*, SC24-6302
- *z/VM: Performance Toolkit Reference*, SC24-6303

RACF® Security Server for z/VM

- *z/VM: RACF Security Server Auditor's Guide*, SC24-6305
- *z/VM: RACF Security Server Command Language Reference*, SC24-6306
- *z/VM: RACF Security Server Diagnosis Guide*, GC24-6307
- *z/VM: RACF Security Server General User's Guide*, SC24-6308
- *z/VM: RACF Security Server Macros and Interfaces*, SC24-6309
- *z/VM: RACF Security Server Messages and Codes*, GC24-6310
- *z/VM: RACF Security Server Security Administrator's Guide*, SC24-6311
- *z/VM: RACF Security Server System Programmer's Guide*, SC24-6312
- *z/VM: Security Server RACROUTE Macro Reference*, SC24-6324

Remote Spooling Communications Subsystem Networking for z/VM

- *z/VM: RSCS Networking Diagnosis*, GC24-6316
- *z/VM: RSCS Networking Exit Customization*, SC24-6317
- *z/VM: RSCS Networking Messages and Codes*, GC24-6318
- *z/VM: RSCS Networking Operation and Use*, SC24-6319
- *z/VM: RSCS Networking Planning and Configuration*, SC24-6320

TCP/IP for z/VM

- *z/VM: TCP/IP Diagnosis Guide*, GC24-6328
- *z/VM: TCP/IP LDAP Administration Guide*, SC24-6329
- *z/VM: TCP/IP Messages and Codes*, GC24-6330

- *z/VM: TCP/IP Planning and Customization*, SC24-6331
- *z/VM: TCP/IP Programmer's Reference*, SC24-6332
- *z/VM: TCP/IP User's Guide*, SC24-6333

Prerequisite Products

Device Support Facilities

- Device Support Facilities (ICKDSF): User's Guide and Reference (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5gc350033/$file/ickug00_v2r5.pdf), GC35-0033

Environmental Record Editing and Printing Program

- Environmental Record Editing and Printing Program (EREP): Reference (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5gc350152/$file/ifc2000_v2r5.pdf), GC35-0152
- Environmental Record Editing and Printing Program (EREP): User's Guide (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5gc350151/$file/ifc1000_v2r5.pdf), GC35-0151

Related Products

z/OS

- *Common Programming Interface Communications Reference (https://publibfp.dhe.ibm.com/epubs/pdf/c2643999.pdf)*, SC26-4399
- z/OS and z/VM: Hardware Configuration Definition Messages (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sc342668/$file/cbdm100_v2r5.pdf), SC34-2668
- z/OS and z/VM: Hardware Configuration Manager User's Guide (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sc342670/$file/eequ100_v2r5.pdf), SC34-2670
- z/OS: Network Job Entry (NJE) Formats and Protocols (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa320988/$file/hasa600_v2r5.pdf), SA32-0988
- z/OS: IBM Tivoli Directory Server Plug-in Reference for z/OS (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa760169/$file/glpa300_v2r5.pdf), SA76-0169
- z/OS: Language Environment Concepts Guide (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa380687/$file/ceea800_v2r5.pdf), SA38-0687
- z/OS: Language Environment Debugging Guide (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5ga320908/$file/ceea100_v2r5.pdf), GA32-0908
- z/OS: Language Environment Programming Guide (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa380682/$file/ceea200_v2r5.pdf), SA38-0682
- z/OS: Language Environment Programming Reference (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa380683/$file/ceea300_v2r5.pdf), SA38-0683
- z/OS: Language Environment Runtime Messages (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa380686/$file/ceea900_v2r5.pdf), SA38-0686
- z/OS: Language Environment Writing Interlanguage Communication Applications (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa380684/$file/ceea400_v2r5.pdf), SA38-0684
- z/OS: MVS Program Management Advanced Facilities (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa231392/$file/ieab200_v2r5.pdf), SA23-1392
- z/OS: MVS Program Management User's Guide and Reference (https://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosv2r5sa231393/$file/ieab100_v2r5.pdf), SA23-1393

XL C++ for z/VM

- XL C/C++ for z/VM: Runtime Library Reference, SC09-7624
- XL C/C++ for z/VM: User's Guide, SC09-7625

Additional Publications

XL C/C++ for z/VM: Runtime Library Reference, SC09-7624
XL C/C++ for z/VM: User's Guide, SC09-7625

Index

Special Characters

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Product Number: 5741-A09

Printed in USA

SC24-6296-73