loadhfs extended (BPX1LDX, BPX4LDX) — Direct the loading of an executable into storage

Function

The loadhfs extended service loads an executable program by path name into the caller's process. This service provides all the functions of "loadhfs (BPX1LOD, BPX4LOD) — Load a program into storage by path name" on page 361 and also allows authorized users to load an executable program into common storage.

Requirements

Authorization:	Supervisor or problem state, any PSW key unless the Lod_Directed flag is specified. When this flag is specified, the caller must be APF authorized, PSW Key 0-7, or Supervisor State.
Dispatchable unit mode:	Task
Cross memory mode:	PASN = HASN
AMODE (BPX1LDX):	31-bit
AMODE (BPX4LDX):	64-bit
ASC mode:	Primary mode
Interrupt status:	Enabled for interrupts
Locks:	Unlocked
Control parameters:	All parameters must be addressable by the caller and in the primary address space.

Format

```
CALL BPX1LDX,(Filename_length,
              Filename,
              Flags,
              Libpath_length,
              Libpath,
              Return_value,
              Return_code,
              Reason_code)
```

AMODE 64 callers need an additional parameter, Entry_point:

```
CALL BPX4LDX,(Filename_length,
              Filename,
              Flags,
              Libpath_length,
              Libpath,
              Entry_point,
              Return_value,
              Return_code,
              Reason_code)
```

Parameters

Filename_length
Supplied parameter

Type: Integer

Length: Fullword

The name of a fullword that contains the length of the Filename parameter. The length can be a value in the range 1 to 1023.

Filename
Supplied parameter

Type: Character string

Character set: No restriction

Length: Specified by the Filename_length parameter

The name of a field that contains the file name of the program that is to be loaded. If the Filename parameter does not contain a slash (/), it is treated as a base name. This parameter 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 the file name is not a base name (that is, it contains at least one slash), the name is used as is; the Libpath parameter is not used to locate the file.

If the file name is a base name, it can be up to 255 characters long.

If the Filename parameter represents a path name, each component of the path name (directory name, subdirectory name, or file name) can be up to 255 characters long. The complete path name can be up to 1023 characters long, and does not require an ending null character.

Flags
Supplied parameter

Type: Integer

Length: Fullword

The Flags parameter is a fullword field. The first three bytes contain option flags. The last byte can be data as defined by an option flag. These constants are defined in the BPXYCONS macro.

Constant	Description
Lod_Directed	Indicates that the target program is to be loaded into the supplied storage subpool. When this option flag is specified, the storage subpool is supplied as the last byte of the FLAGS parameter. This flag is only supported for authorized system callers (APF authorized or system key or supervisor state). Unauthorized callers specifying this flag receive a EPERM error return code. When this flag is specified, it is the responsibility of the caller to free the program storage. Only subpool 241 is currently supported; any other subpool specified results in an EINVAL error return code. The storage obtained for the target program is key 0 storage. Lod_Directed takes precedence over Lod_Ignore_Sticky, which in turn takes precedence over Lod_Error_St_ExLink.
Lod_Error_St_ExLink	Indicates that LOAD processing is to be bypassed if the file is an external link or has the sticky bit set on. If the file has the sticky bit set or is an external link, the request fails with return code EPERM (the operation is not permitted) and a reason code of JrExternalLink or JrStickyBit.

Constant	Description
Lod_Ignore_Sticky	Indicates that the sticky bit for a file is to be ignored. If the file is sticky, it is loaded from the z/OS UNIX file system.

Note: If both Lod_Ignore_Sticky and Lod_Error_St_ExLink are specified, the Lod_Ignore_Sticky option is honored, and Lod_Error_St_ExLink is ignored.

Libpath_length
Supplied parameter

Type: Integer

Length: Fullword

The name of a fullword that contains the length of the library path parameter. If a value of zero is specified, the library path parameter is ignored.

Libpath
Supplied parameter

Type: Structure

Length: Specified by the Libpath_length parameter

The name of a field that contains the library path to be searched to determine the fully qualified path name of the file that is specified. The library path can contain 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 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 1021 bytes.

The following is an example of a valid library path:

- **/usr1/bin:/grp1/bin:/bin**

Entry_point
Returned parameter (BPX4LDX only)

Type: Structure

Length: Doubleword

The name of a field in which either an entry point address or the address of a structure is returned. If the Lod_Directed flag is specified, this service returns the address of a 24-byte structure that contains the length of the loaded program storage, followed by the start address of the loaded program, followed by the entry point address of the loaded program. The returned structure is mapped in the BPXYCONS macro.

Return_value
Returned parameter

Type: Integer

Length: Fullword

The return value for this service is as follows:

- For an AMODE(31) caller, the name of a fullword in which the loadhfs extended service returns -1 if it is not successful. If it is successful, the loadhfs extended service returns the entry point address of the program that was loaded into storage, unless the Lod_Directed flag is specified. If the

Lod_Directed flag is specified, this service returns the address of a 24-byte structure that contains the length of the loaded program storage, followed by the start address of the loaded program, followed by the entry point address of the loaded program. If the loaded program is an AMODE(31) program, the high-order bit of the entry point address is ON. The returned structure is mapped in the BPXYCONS macro

- For an AMODE(64) caller, the Return_value is returned as either `0` if successful or `-1` if not successful.

Return_code
Returned parameter

Type: Integer

Length: Fullword

The name of a fullword in which the loadhfs extended service stores the return code. The loadhfs extended service returns Return_code only if Return_value is –1. See *z/OS UNIX System Services Messages and Codes* for a complete list of possible return code values. The directed loadhfs service can return one of the following values in the Return_code parameter:

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; 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.
EAGAIN	The file changed during load processing (JrFileChangeDuringLoad).
EINVAL	An invalid parameter value was specified. The invalid parameter can be Filename_length, or FLAGS. If FLAGS is incorrect, a reason code of either JrOptionFlagsErr (unsupported FLAGS parameter value), or JrLodDirectedSubpoolError (unsupported value for the directed loadhfs subpool passed in the FLAGS parameter).
ELOOP	A loop exists in symbolic links that were encountered during resolution of the Filename parameter. This error is issued if more than 24 symbolic links are detected in the resolution of Filename.
EMVSERR	An error occurred while loading a z/OS UNIX program (JrMVSLoadFailure or JrMVSPgmNotFound). Or an error occurred checking the caller's environment against the authorization of the file (JrNoListAuthPgmPath, JrNoListPgmCntlPath, JrProgCntl, JrAuthCaller).
ENAMETOOLONG	The Filename parameter is longer than 1023 characters; or some component of the file name is longer than 255 characters. Name truncation is not supported.
ENOENT	No file name was specified, or one or more of the components of the specified Filename parameter were not found.
ENOEXEC	The specified file has execute permission, but it is not in the proper format to be a process image file.
ENOMEM	The file that is to be loaded requires more memory than is permitted by the hardware or the operating system, or a storage request failed for the directed load target (JrLodDirectedNoStorage).
ENOTDIR	A directory component of the Filename parameter is not a directory.

Return_code	Explanation
EPERM	The operation is not permitted. The Flags parameter was set to Lod_Error_St_ExLink. If the file has the sticky bit set or is an external link, the request fails with reason code of JrStickyBit or JrExternalLink, respectively. Or an unauthorized caller specified the Lod_Directed option flag (JrLodDirectedAuthErr).

Note: In addition to the return codes listed here, the loadhfs extended service can return additional errors for other failures that can occur on a stat or an open syscall.

Reason_code
Returned parameter

Type: Integer

Length: Fullword

The name of a fullword in which the loadhfs extended service stores the reason code. The loadhfs extended service returns Reason_code only if Return_value is –1. Reason_code further qualifies the Return_code value. For the reason codes, see *z/OS UNIX System Services Messages and Codes*.

Usage notes

Note that Usage Notes 1–9 do not apply if you specify the Lod_Directed flag.

1. A prior loaded copy of a z/OS UNIX program is reused under the same circumstances that apply to the reuse of a prior loaded MVS unauthorized program from an unauthorized library by the MVS LOAD service, with the following exceptions:
   - If the calling process is in Ptrace debug mode, a prior loaded copy is not reused.
   - If the calling process is not in Ptrace debug mode, but the only prior loaded usable copy of the HFS program found is in storage that is modifiable by the caller, the prior copy is not reused.
2. If the specified file name represents an external link or a sticky bit file, the program is loaded from the caller's MVS load library search order. For an external link, the external name is only used if the name is eight characters or less, otherwise the caller receives an error from the loadhfs service. For a sticky bit program, the file name is used if it is eight characters or less. If the file name is greater than eight characters, or the MVS program is not found, the program is loaded from the z/OS UNIX file system.
3. When it is running from a pthread_created thread (pthread), the specified file is loaded into storage and associated with the Initial Pthread Creating Task (IPT). This allows the program to be shared across multiple threads, without the problem of its disappearing unexpectedly when a thread terminates.
4. When the calling process is being debugged via the ptrace service, the following applies:
   - Programs that are loaded using this service are loaded into storage that is modifiable by the caller of the loadhfs service.
   - A call to this service generates a WastStopFlagLoad Ptrace event to the debugger process.
5. 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. These flags have meaning only for an execed or spawned program.

6. Because the z/OS UNIX file system is not an authorized library, the following restrictions apply:
   - Loading a program from the z/OS UNIX file system causes the program environment to become uncontrolled unless the executable file has the program control attribute turned on (ST_PROGCTL). Not having the program control attribute on prevents future invocations of authorized programs like PADS programs. In addition, PADS programs should not attempt to load programs from the z/OS UNIX file system; the z/OS UNIX file system is considered an unauthorized library and can potentially be modified by users that do not have the same level of authorization as the PADS program.
   - System key, supervisor state and APF-authorized callers receive an EMVSERR with reason code JrAuthCaller if the caller attempts to load a program from the z/OS UNIX file system, unless the executable file has the APF attribute turned on.
7. If a program that is loaded into storage with this service is not deleted from storage, the program remains in storage until the calling task terminates, if it is not a pthread. If the caller is a pthread, the program remains in storage until the Initial Pthread Creating Task (IPT) terminates.
8. The AUTHPGMLIST system parameter applies to this system call. AUTHPGMLIST specifies a z/OS UNIX file that contains a list of sanctioned directories or authorized program names. If activated, an additional level of security checking will be performed to ensure that the program being loaded is coming from an authorized directory in the z/OS UNIX file system or is an authorized MVS program name. For details about the sanction list, see the topic on using sanction lists in *z/OS UNIX System Services Planning*.
9. The following apply to shared program libraries:
   - Executables that have the ST_SHARELIB extended attribute turned on are considered system shared library programs. System shared library programs are the most optimal way to share large executables across many address spaces in the system. These executables are shared on a megabyte boundary to allow for the sharing of a single page table (similar to LPA). The storage used in the user address space to establish the mapping to the shared library region is from the high end of private storage.
   - If the program to be loaded is determined to be a shared library program (that is, if the ST_SHARELIB extended attribute is on), the loadhfs service queries the shared library region to determine if the target program is there.

     When a shared library program is loaded anew into the shared region or reloaded from the shared region, the program is mapped from the shared region into the private area of the calling address space. It is important to note that, because the program is not actually reloaded from DASD into the private area of each calling address space, but only remapped from the shared region, shared library programs are more efficient in their utilization of system resources than normal private area programs. For this reason, programs that are to be shared across several address spaces in the system are good candidates for identification as shared library programs.

     If a target program is not in the shared library region and cannot be loaded into the region because of its attributes, the program is treated like a private area program and is loaded into the caller's private area storage.

     Additionally, if the calling address space cannot accommodate the target address for the shared library program, the program is treated like a private area program.

- In order for a program to be honored as a shared library program, certain conditions must be met:
  - The program must be a z/OS UNIX program module; MVS library modules cannot be loaded into the shared region.
  - A sticky bit program that is found in the MVS search order is not honored as a shared library program.
  - The program cannot be a multiple-segment (split RMODE) load module; multiple-segment load modules are not supported in the shared library region.
  - The program must have read “other” permission and be link-edited as REENTRANT.
- A shared library program can reside in a file system that was mounted with the NOSETUID operand.

10. When the Lod_Directed flag is specified:
    - It is the responsibility of the caller to manage the storage associated with the loaded program. When Lod_Directed is specified, deletehfs can not be used to remove the executable from storage. The executable will stay in storage until freed. The storage can be freed using the returned storage length and program start address.
    - It is the responsibility of the caller to use the CSVDYLPA ADD BYADDR(YES) service to create a CDE in order to provide serviceability information for the loaded program. Without this, serviceability functions, such as SLIP LPAMOD and IPCS WHERE, are not available for the loaded program.
    - The caller must save a copy of the returned program information after each call. The returned data structure is reused for each syscall by a given task. The returned program information structure is cleared if the call is made and an error occurs.
    - A program loaded with the Lod_Directed flag can not be debugged using Ptrace debug mode.
    - The shared library program attribute, st_Sharelib, is ignored.
    - The sticky bit for a file is ignored whether or not Lod_Ignore_Sticky is specified.
    - If the file is an external link, the request will fail with return code of EPERM (the operation is not permitted) and a reason code of JrExternalLink whether or not Lod_Error_St_ExLink is specified.

Related services

None.

Characteristics and restrictions

There are no restrictions on the use of the loadhfs extended service.

Examples

For an example using this callable service, see “BPX1LDX (loadHFS extended) example” on page 1301.