# PIPEDDR
# Dump/Restore a disk using CMS Pipelines

## Description

The PIPEDDR EXEC can dump a disk into a regular CMS file, to an ftp server, to a file in an NFS share (which is mounted to BFS, the CMS Byte File System), or via TCP/IP directly to a disk on a remote system. Restore a disk from a PIPEDDR output file via one of the same methods. The data is packed or can also be compressed further, saving network bandwidth or disk space. The data can also be encrypted either on disk or as it is sent over the network. A remote disk can be kept in sync with a local disk using the remote update function.

The PIPEDDR EXEC can save and restore disk images using either the raw disk I/O support available in CMS Pipelines or the Pipelines interface to the CMS DDR command. The internal structure of each kind of disk image are different and are not compatible with each other. The Pipelines raw disk I/O support uses the Pipe stages TRACKREAD and TRACKWRITE (for ECKD disks) or FBAREAD and FBAWRITE (for FB-512 disks.) These stages are part of either z/VM® 6.4 and later or the optional “Runtime Distribution” of CMS Pipelines. The interface to the CMS DDR command uses the DDR Pipelines stage. This Pipelines stage amd the updated DDR module are included with z/VM 6.2 and later. If you are on an older release of z/VM, then you can get the required files from the DRPC package in the VM download library.

The exec dumps an entire disk into a regular CMS file. The operating system format of the disk or the type of data on it does not matter. The disk is read in binary track by track or block by block. The output data by default is written in packed format compatible with the PACK option of the CMS COPYFILE command and is in fixed record format with 1024 byte records. This allows easier interchange of the backup files with non-mainframe systems. These files can be used for backups of disks or restored on another system. There are also options to compress the output to make the file even smaller. See the notes section of the help file for more information on compressing the output.

The output file created by this exec has meta data about the source disk as the first record. The contents of the disk follow in the rest of the records. This ensures that the disk is restored to a device of the same type and size. Data must be restored to the same type of device using the same disk data encoding method. In other words, ECKD devices (3390) must be restored to ECKD devices and FB-512 devices must be restored to FB-512 devices. It is not possible to convert the data on one device type to another device type using this exec.

The first record also contains data about the id that dumped the disk, what kind of compression was used, if a data verification code such as a CRC or digest value is included, and if the disk data is encrypted. This allows the exec to correctly restore the data.

The disk can also be transferred over a TCP/IP connection directly to a receiving PIPEDDR exec listening to an IP port on a remote system. The default is to send the entire disk. If the disk was transferred previously, a PIPEDDR option will send just the changes to the blocks or tracks to the remote system instead of the entire disk. Recent updates to CMS Pipelines and TCPIP allow optional secure (TLS) remote sessions. It is recommended that the same version of the PIPEDDR exec be used on both ends of the TCP/IP connection. PIPEDDR also supports sending the output file directly to an ftp or ftps (secure ftp) server. The file created on the ftp server is in the same format as the CMS file. A disk can be restored from a file hosted by either an ftp server or an HTTP (web) server. Secure connections are supported.

Byte File System (BFS) paths are also supported, which allows the file of the disk data to be hosted by a remote NFS server. You must mount the remote NFS share into the BFS filesystem to use this support. Please see the usage note in the help file about BFS and NFS for more information.

Dump to and restore from a tape is supported by using the TAPE output method and the same operations using a CMS filedef using the FILEDEF output method.

By default, PIPEDDR will display messages on the console about the progress of the disk operation as it processes a disk. The default behavior is to display a message for every 10% of disk that has been processed. No messages are displayed for small disks and more messages may be displayed for larger disks. The display can be suppressed by using the NOTYPE option or changing the default using the PIPEDDR SET command.

If the disk contains sensitive data, the disk contents can be encrypted, using the CPACF hardware encryption processor. You must obtain the *KM Package* from the VM download library to enable encryption. Find it at *http://www.vm.ibm.com/download/packages/descript.cgi?KM*. The KMRTNS CSLLIB is used from the *VMARC file*. These CSL routines require your machine to have the CPACF feature enabled to use the built in hardware encryption and decryption.

The raw disk I/O interface requires either z/VM 6.4 and later or the *Princeton Runtime Distribution* level of Pipelines (found at *http://vm.marist.edu/%7Epipeline/index.html#Runtime*.) at least version 110B0004 (15 May 2002 level) for reading and writing ECKD disks and level 110C0004 (01 Jul 2010 level) or later for reading and writing FBA disks. PIPEDDR uses the PICKPIPE EXEC to load the runtime distribution level of Pipelines if it is needed and if it is available. PICKPIPE is also available via the VM download library. See the *PICKPIPE description* or get the *VMARC file*.

The Pipelines interface to CMS DDR requires either z/VM 6.2 or later or the DRPC and DDR modules from the DRPC package on the VM download library. See the *DRPC description* or get the *VMARC file* to obtain this function. The updated level of CMS Pipelines is not required for CMS DDR support.

Dumping or restoring to FTP supports 3 different FTP pipeline stages. The newest ftp stage is built in to Pipelines and is available if secure TCP/IP connections are supported by CMS Pipelines. (APAR VM66365 for z/VM 7.1, included in z/VM 7.2 and later.) This stage supports both secure and unsecured ftp sessions. The other ftp stages supported by PIPEDDR are not built-in to Pipelines; they are loaded from a separate module or file. The preferred external ftp stage is found in the FTPREXX package on the VM downloads page. Get the *FTPREXX Package* to download this stage.

The final one is supplied with z/VM, but only supported for the installation of z/VM. It requires either the INSTPIPE MODULE from a modern version of CMS (found on MAINTvrm 4CC or MAINT 193) or the DRPC MODULE. See the reference to the DRPC MODULE in the previous paragraph about CMS DDR.

Transfers to and from FTP will use a NETRC DATA file stored anywhere in the search order. This file can supply a password or a userid and password for the connection, so that the password would not be hardcoded into an exec or displayed on the console.

The exec also supports two other forms of compaction, TERSE and ZLIB. The TERSE option can only be used if the TERSE Pipelines stage is available. This allows the exec to create output files that are smaller or send less data over the network. The TERSE Pipelines stage is part of the PIPSYSF filter package which is now available from the Marist Pipelines page. See the *PIPSYSF Filter Package* page for more information and a link to download the module.

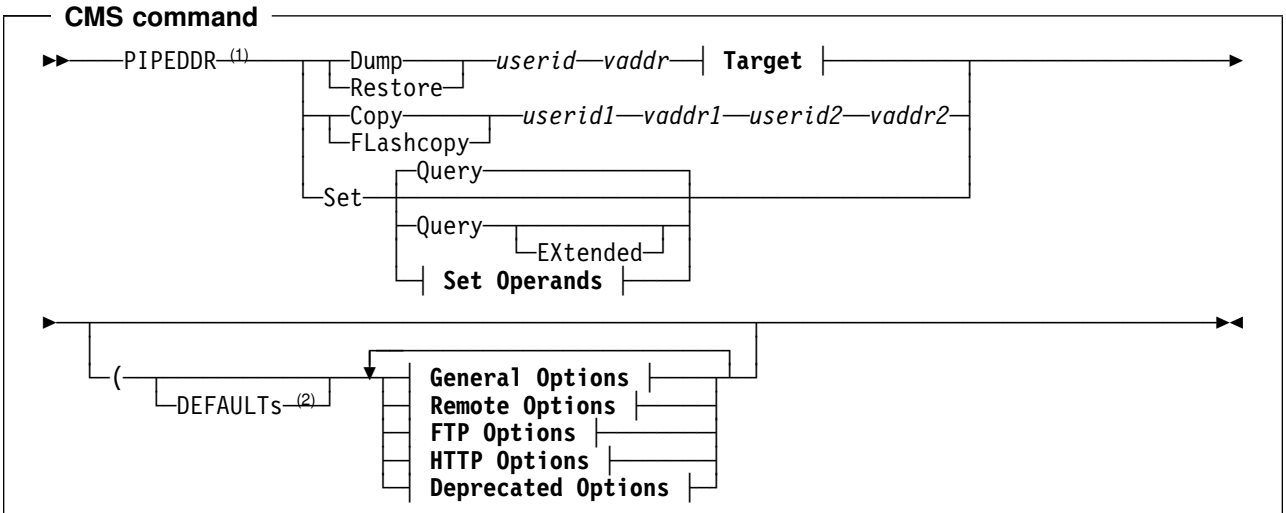
The ZLIB compression option requires the zlib Pipeline stage which is found in in the ZLIBSTG MODULE file. You can download the file "fplgcc.vma" which is a VMARC format file downloadable from *John Hartmann's space* on GitHub. The URL is *https://github.com/jphartmann/cmslib-exec*.

If the CMS DDR format is used, the DDR module supports 2 compression options, COMPRESS and LZCOMPACT, which may provide an acceptable level of compression.

There is also a copy option which preforms a disk to disk copy, with automatic linking of the source and target disks. If the flashcopy option is specified, the exec attempts to copy the disks using the CP FLASHCOPY command. If that command fails or your user id is not allowed to use this command, the exec falls back to a disk to disk copy. The copy option is no different than copying a disk using DDR, and in fact just the DDR module is used if the CMSDDR option is specified on the copy command.

A help file is included in the package and it contains a detailed description of all of the options. Enter HELP PIPEDDR on the CMS command line for usage information.

## Syntax:

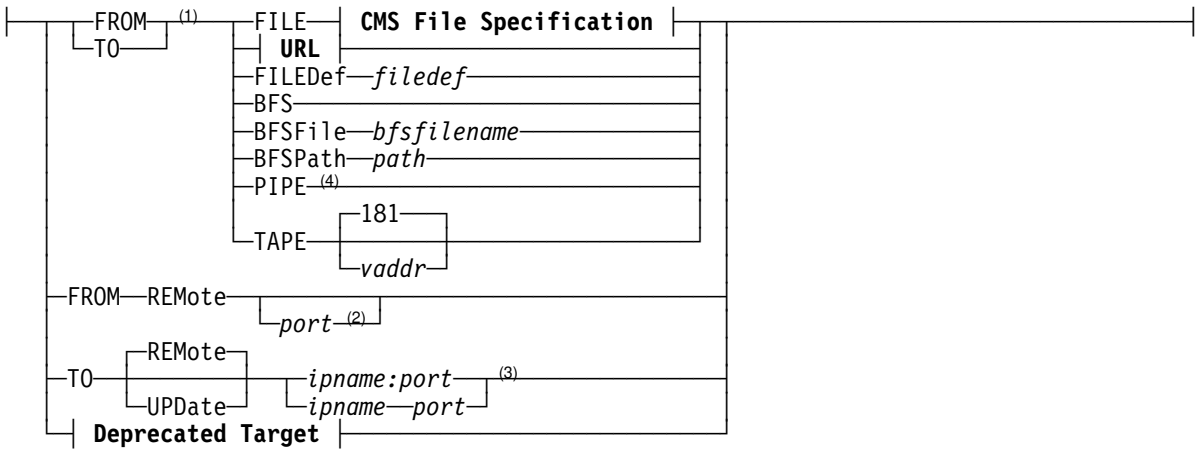



**Notes:**

1 Uppercase letters denote the shortest acceptable abbreviation. If a keyword appears entirely in uppercase letters, it cannot be abbreviated. Options, except for DEFAULTS, can be specified in any order.

2 This option, if used, must be specified before any other options.

**Target**




**Notes:**

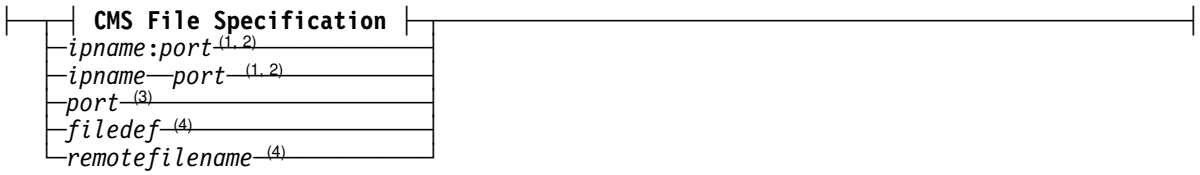
1 Keyword TO is used with Dump commands and keyword FROM is used with Restore commands.

2 The port can be specified here but the PORT option is preferred.

3 The port number is omitted here if the PORT option is specified.

4 The PIPE keyword is only valid and is the default when PIPEDDR is called as a Pipelines stage.

**Deprecated Target**




**Notes:**

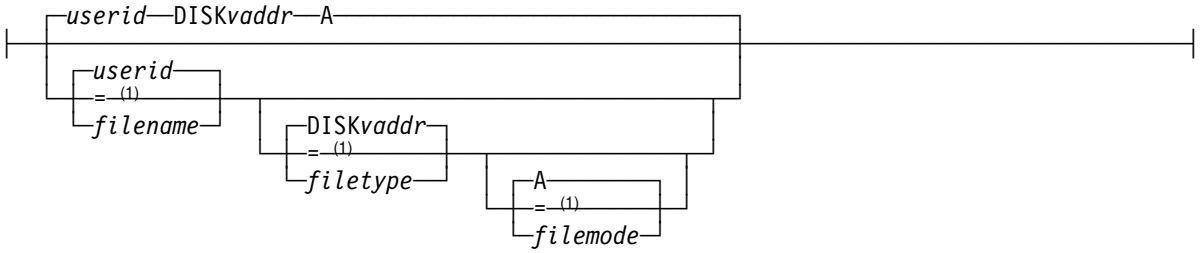
1 The TO and REMOTE keywords must be specified if the ipname does not contain a period or is the same as another keyword.

2 The port can be omitted if the PORT option is specified.

3 The deprecated LISTEN option allows the port to specified here but the PORT option is better and FROM REMOTE is preferred.

4 Only valid if one of the deprecated options FTP, HTTP, BFS, or FILEDEF is used.

**CMS File Specification**

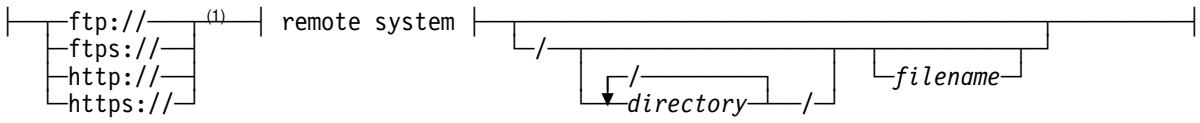



**Note:**

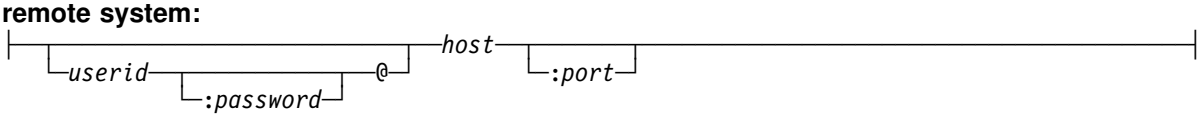
1 The equals sign means to use the default value for this positional part of the file specification.

## URL

**url:**



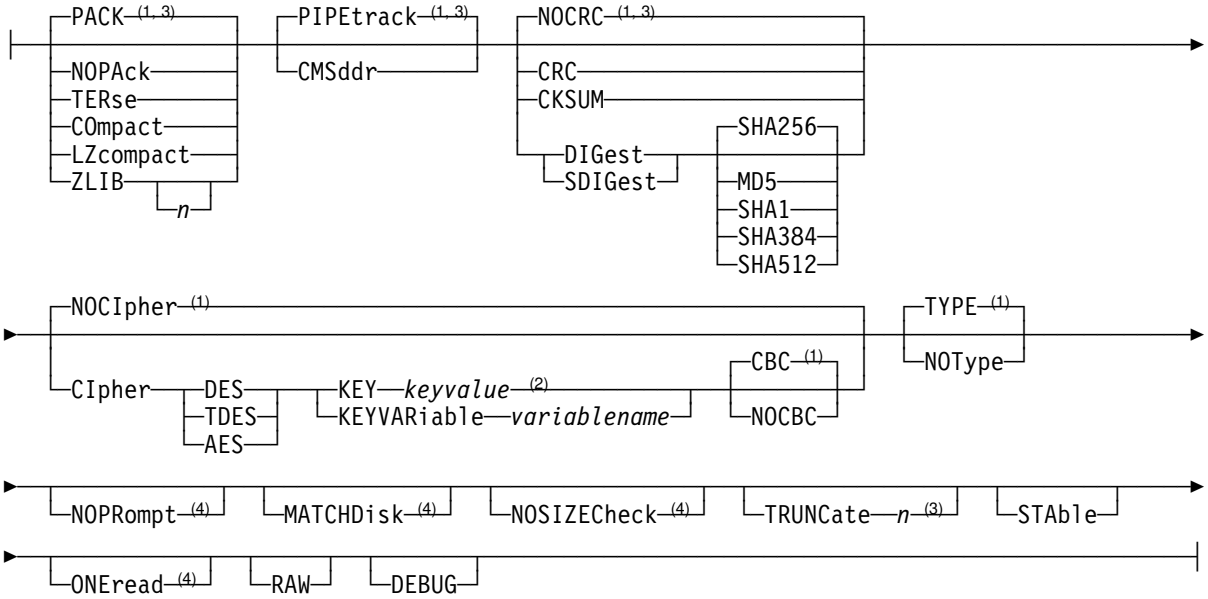

**remote system:**




**Note:**

1 Do not enter any spaces when entering a url.

## General Options




**Notes:**

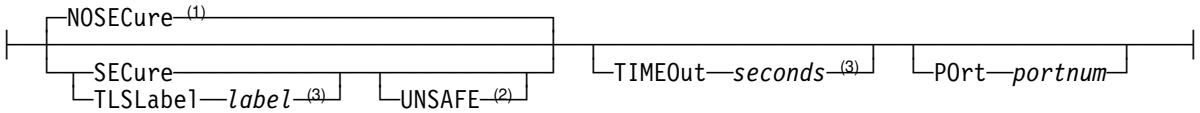
1 The default for this option may change if the SET command is used to specify a different default.

2 The key can be omitted if the key was stored with the SET command.

3 This option is only specified with a Dump command; it is ignored on a Restore or Copy command.

4 This option is only specified with a Restore command; it is ignored on a Dump command.

**Remote Options**




**Notes:**

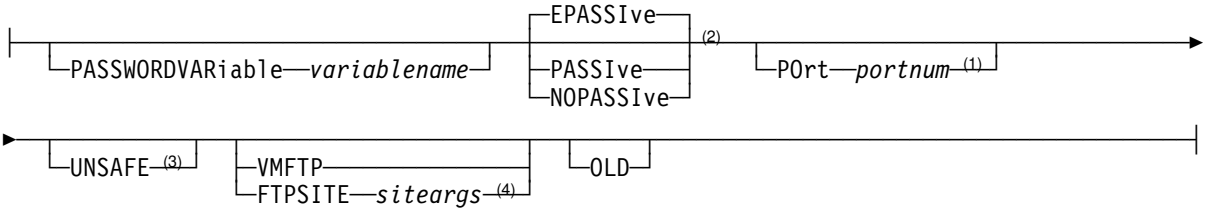
1 The default for this option may change if the SET command is used to specify a different default.

2 This option is only specified with a Dump command; it is invalid or ignored on a Restore command.

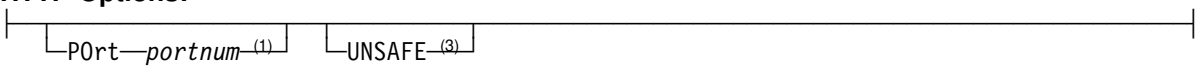
3 This option is only specified with a Restore command; it is invalid or ignored on a Dump command.

**FTP and HTTP Options**

**FTP Options:**




**HTTP Options:**




**Notes:**

1 The PORT option is allowed, but a port number in the URL has priority.

2 These options are only valid if the FTP REXX stage is used.

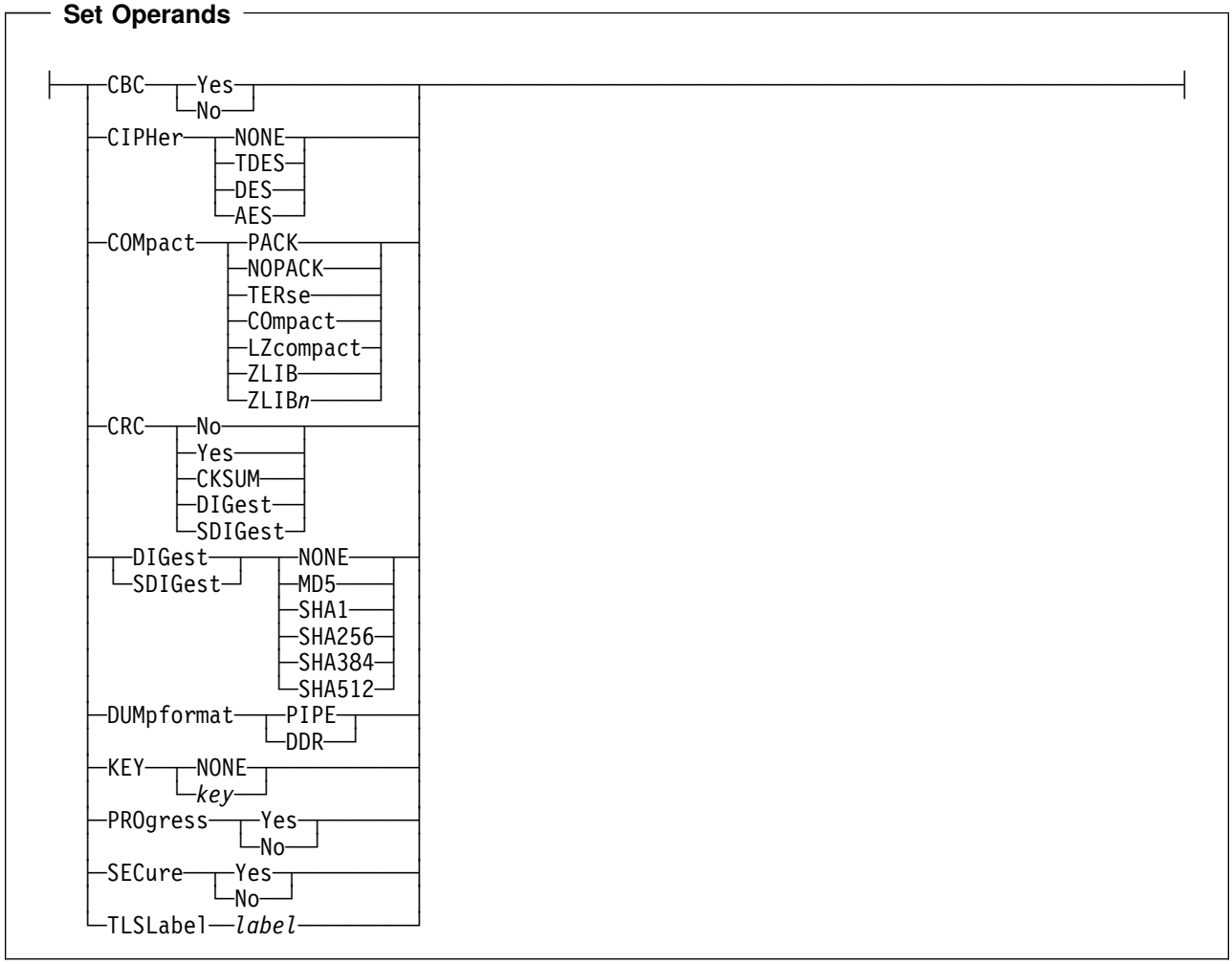
3 This option only applies to secure ftp or http connections.

4 A site command with multiple words can be specified. See the description of FTPSITE command for more information.

**Deprecated Options**


**Note:**

1 These options still function, but the **Target** operands are recommended.




## Operands:

**Dump**
**Restore**

Dump or Restore a disk. The Dump function only reads the disk. The Restore function overwrites all of the data on the disk.

*userid vaddr*

specifies the disk to dump or restore. Specifying a user ID of "*" (an asterisk) will use an existing linked or attached disk on your user ID. If you dump or restore a disk owned by another user ID, you must have permission to link the disk or know the appropriate password. Disks are linked read only for dumps and read/write for restore.

**Copy**

Make a local copy of a disk, using either the DDR module or Pipelines.

**FLashcopy**

Make a local copy of a disk, using the CP FLASHCOPY command. If this command is not available or it fails, the DDR module is used to copy the disk.

*userid1 vaddr1*

specifies the input disk to be copied.

*userid2 vaddr2*
specifies the output disk of the copy.

**Set**
Set a program default which is stored as a global variable. See "SET Command Operands" on page 10 for details about the defaults that can be selected.

**Query**
**Query EXtended**
Display the current settings. This is the default if no operand is specified on a **Set** command. **QUERY** shows all settings except a stored cipher key. **QUERY EXTENDED** shows all settings, including the key.

## Target

This section describes the target that the disk data is read from or written to.

**FROM**
**TO**
Keywords that describe where the disk data is coming **FROM** (for a Restore command) or going **TO** (for a Dump command.) The **FROM** or **TO** keywords can be omitted for some targets so that the exec remains compatible with its older syntax. However, using these keywords is recommended.

**FILE**
Keyword that indicates that a CMS file specification follows.

*filename*
*filetype*
*filemode*
The CMS file specification for the input or output file. If the filename is omitted or specified as "=", it defaults to the user ID that owns the disk. If the filetype is omitted or specified as "=", it defaults to the keyword DISK followed by the 4 digit virtual device address of the input disk (DISK*nnnn*). If the filemode is omitted or specified as "=", it defaults to A. The file specification can also be specified as "= = fm" (without the quotes) to use the default file naming but with the specified CMS filemode for input or output.

**FROM REMote**
The contents of an entire disk are received from a remote system that is also running the PIPEDDR EXEC. The exec waits for an incoming TCP/IP connection from a remote system running a PIPEDDR Dump command. If a TCP/IP port is not specified using the **PORT** option, or with the positional arguments to PIPEDDR, the next available port is obtained and the listening port number is displayed on the console. PIPEDDR will keep waiting for a connection until one is established or the pipeline is stopped. If you want PIPEDDR to only wait a limited amount of time, specify the **TIMEOUT** option.

**TO REMote**
The contents of the disk are sent to a remote system that is also running the PIPEDDR EXEC. The remote system is specified by the *ipname* and *port* arguments. PIPEDDR must already be running and waiting for a connection on the remote system. You can specify other general options such as compression or encryption selections to specify how the data is processed before it is sent over the network. If only the keyword **TO** is specified, then the **TO REMOTE** function is assumed.

**TO UPDate**
The contents of the remote disk are compared to the local disk using hash values and only changed tracks or blocks are sent ("updated") on the remote system. The remote system must be running PIPEDDR with the **FROM REMOTE** function specified and waiting for the connection.

See “Usage Note 11. More about incrementally updating a remote disk” on page 24 for more information about this function.

*ipname*
*ipname:port*
*ipname port*
an IP name or address and optionally a port of a remote system. If a port is specified, it is separated from the IP name using a colon or a blank. If the **PORT** option is used, a port that is specified here as part of the IP name is ignored. If the IP name does not contain any periods, then the keywords **TO REMOTE** are required, otherwise the value is assumed to be a file name.

**FILEDef** *filedef*
*filedef*
Dump or restore a disk to a CMS Filedef. The filedef must be defined with the CMS FILEDEF command before PIPEDDR is invoked. Any type of filedef can be defined that is usable on the Pipelines “qsam” stage. Any error messages shown when using this method are from CMS Pipelines. The default output record format is Fixed 1024 byte records. The **ONEREAD** option is implied when using this method with a **Restore** command. If the **FILEDEF** keyword is not specified, then the deprecated option FILEDEF must be used to indicate that the argument is a filedef name and not the name of a file.

*remotefilename*
a file specification on a remote system. Mixed case and long filenames are allowed. This argument is only assumed to be a file specification if one of the deprecated options BFS, BFSPATH, FTP, or HTTP is specified. If one of these options is used and the remote file name is omitted, it defaults to *userid*.DISK*nnnn* with all uppercase characters converted to lower case. If the ftp or http option is specified and a filename (-f or -filename option) is specified in the parm string or in the ftp or http URL, the *remotefilename* is ignored. Otherwise, if one of the ftp, http, or bfs options is specified, the file specification is taken “as is” and passed to the remote system as the file to dump to or restore from.

*url* a url specification. An HTTP url is only valid for input and an FTP url is valid for input and output. See “Usage Note 3. More about the URL argument” on page 19 for more information.

**BFS**
**BFSFile** *bfsfilename*
**BFSPath** *path*
The input or output file resides in BFS (the CMS Byte File System) or in an NFS share that is mounted into the BFS filesystem.

If the argument is **BFS**, then the input or output file will be the default name of *userid*.DISK*nnnn* with all uppercase characters converted to lower case. It is read from or stored into the current working directory set via the OPENVM SET DIRECTORY command.

To specify the file name, use the **BFSFile** argument to specify the name. The *bfsfilename* can also include the path to the file. The normal rules for paths in Posix file specifications apply; if the path starts with a slash (“/”), then it specifies an absolute path; if it does not, then the path is relative to the current working directory.

The **BFSPath** argument specifies the path to an input or output file that has the default name. The path can be a relative or absolute path to the file.

If the *bfsfilename* or the *path* contain blanks, the name or path may be enclosed in quotes. Either a single quote (') or double quote (") can be used. Quotes can be specified as part of the name or path in the same way they are specified in Rexx strings. The processing of quotes is not extensive. If the string begins with a quote, PIPEDDR assumes it also ends with one. If you see an error message from CMS pipelines, check how you have specified the file name or path.

See “Usage Note 5. Using BFS and NFS files for input and output” on page 21 for more information. Synonyms for these arguments are **HFS** instead of **BFS**, **HFSFile** instead of **BFSFile**, and **HFSPath** instead of **BFSPath**.

**PIPE**
The PIPEDDR EXEC is invoked as an argument to the *rexx* CMS Pipelines stage. When the argument is **DUMP**, the disk data is output to the pipeline. When the argument is **RESTORE**, the disk data is received as input from the pipeline. Only the DUMP and RESTORE arguments are valid.

When the argument is **DUMP**, PIPEDDR should be the first stage in the pipeline segment. When the argument is **RESTORE**, PIPEDDR should be the last stage in the pipeline segment. It does not pass any records to its primary output stream. PIPEDDR supports a secondary output stream for both DUMP and RESTORE. If this stream is connected, progress messages are output to this stream instead of being shown on the console. No messages are issued if the NOTYPE option is specified or for a small input or output disk.

Here is an example of using PIPEDDR as a pipelines stage:

```
PIPE rexx (pipeddr exec) dump maint 190 to pipe (digest | count bytes | console
```

**TAPE**
Dump a disk to tape or restore a disk from tape. A tape drive virtual address can be specified after the **TAPE** keyword. If no address is specified, it is assumed the tape drive is attached at virtual address 181. The disk must fit on 1 tape device; multiple tape drives are not supported by this option. The **FILEDEF** keyword can be used instead along with the multi-volume tape support available in CMS. The tape drive is set to compress mode before writing the data using the command “TAPE MODESET (COMP”.

## SET Command Operands

This section describes the valid values for each Set option.

**CIPHer**
Sets the default to the specified cipher method. Specifying NONE sets the default to NOCIPHER. If a default cipher key has not been specified, then the cipher key must be included as a command option.

**CBC**
Sets the default cipher mode. If CBC is Yes or not specified, the default cipher mode is CBC. If CBC is No, the default mode is ECB. See “Usage Note 10. More about the encryption options” on page 23 for more about the cipher mode.

**COMpact**
Sets the default compaction option.

**CRC**
Sets the default for the type of verification used to ensure that the disk data has not been altered. A value of Yes sets the default to use the CRC option. A value of No sets the default to no verification (the NOCRC option.) A value of CKSUM sets the default to the CKSUM option. A value of DIGEST or SDIGEST set the default to one of those options. The digest type can be set using the DIGEST or SDIGEST set options. If no digest type is set, the default digest of SHA256 is used.

**DIGest**
**SDIGest**
Set the type of verification (the CRC setting) to be the specified keyword and sets the default digest type. If no default digest type is set, the default type is SHA256. If the CRC setting is changed to one of the non-digest settings of YES, NO, or CKSUM, the digest type is changed to NONE.

**DUMpformat**
Sets the default dump format. A value of PIPE sets the default to use the Pipeline track read/write stages for ECKD disks or the fba stages for FBA disks. A value of DDR sets the default to use the CMS DDR program.

**PROgress**
Sets the default for the display of progress messages, which are the TYPE and NOTYPE options on the command. A progress value of Yes sets the default option to TYPE. A value of No sets the default to NOTYPE.

**KEY**
Sets the default cipher key. The keyword NONE removes the stored key. The key can be specified as a hexadecimal string, a single word, or a phrase enclosed in parentheses or quotes (single or double.) The key is stored as a hex string in GLOBALV. Note that any user that has access to the LASTING GLOBALV file can read this key. Also see “Usage Note 10. More about the encryption options” on page 23 for information about storing a key.

**SECURE**
Sets the default for secure remote connections. YES sets the default to SECURE and NO sets it to NOSECURE.

**TLSLabel**
Sets the TLSLABEL used for receiving disk images over secure remote connections. If this value is set, the SECURE setting is also set to YES. Specifying a blank label will clear the stored label value. A label must be 1 to 8 alphanumeric characters.

# Options:

## General Options

These options apply to all input and output selections.

**DEFAULTs**
Ignore any program defaults stored using the **Set** operand for this execution of the exec. All built-in program defaults are used unless another option is specified that overrides the default. This option is useful when PIPEDDR is invoked in an exec or in automation so that the output settings are known and not changed by saved options. If this option is specified, it must be the first word of the options string.

**PACK**
Compress the dumped disk contents using the PACK Pipeline stage. The format of the file is compatible with the PACK option of COPYFILE. This is the default compression method unless a different default has been specified using the PIPEDDR SET command.

**NOPAck**
Do not pack the output file when the disk is dumped. The record structure of the data is preserved by adding a record descriptor word (rdw) to the beginning of each record. If the output is written to a CMS file, the file may be fixed 1024 byte records or variable, depending on what other options are specified.

If the RAW option is also specified, no blocking or packing is performed on the data. The intent of this option is for testing and to be able to view the disk data in an unpacked file. If the input device is FBA, the output file is created in fixed record format because the logical record length is too large for a CMS variable format file. The last record is padded with binary zeros; if PIPEDDR is not used to restore the disk image, an error message may be issued because of this padding.

**TERse**
Compress the dumped disk contents using the TERSE Pipeline stage. This stage is part of the PIPSYSF filter package that must be downloaded from the Pipelines Runtime Distribution page. The format of the file is compatible with the TERSE, FCOPYTRS, and DETERSE modules. See "Usage Note 6. Notes about compression" on page 21 for more information about compressing data and alternatives if the TERSE option is not available to you.

**COMPact**
Compress the dumped disk contents using the COMPACT output option of the CMS DDR command. This option is only valid if the CMSDDR option is also specified or DDR format dumps are the default.

**LZcompact**
Compress the dumped disk contents using the LZCOMPACT output option of the CMS DDR command. This option is only valid if the CMSDDR option is also specified or DDR format dumps are the default.

**ZLIB** *n*
Compress the dumped disk contents with the "compress" algorithm using the ZLIB pipelines stage. This stage is available in the ZLIBSTG MODULE. See item 3 under "Usage Note 6. Notes about compression" on page 21 for more information on how to obtain this module. The optional parameter "*n*" specifies the compression level from 1 which is no compression to 9, which is the most aggressive. If no compression level is specified, the default value that is built-in to the module is used. To specify a default compression value on a Set command, leave out the space between ZLIB and the number. For example PIPEDDR SET COMP ZLIB5

**PIPEtrack**
Use Pipelines format for input and output. For ECKD disks, the trackread and trackwrite stages are used. For FBA disks, fbaread and fbawrite are used. These stages require the updated CMS Pipelines runtime module if your z/VM system is older than z/VM 6.4. See "Usage Note 1. Disk input/output formats" on page 18 for more information. Other synonyms that are accepted for this option are PIPEddr and PIPEfba. This format is the default unless a new default is specified using the PIPEDDR SET DUMPFORMAT command.

**CMSddr**
Use CMS DDR format for input and output. This support requires the DRPC and DDR modules that support the Pipelines interface to DDR. The updated modules are a standard part of CMS in z/VM release 6.2 and later. For older releases, the support is available from the DRPC package available from the VM download library. See "Usage Note 1. Disk input/output formats" on page 18 for more information. The CMSDDR option is also assumed if the exec is renamed to CMSDDR EXEC or VMDDR EXEC, or the exec is invoked via a synonym named CMSDDR or VMDDR.

**TYPe**
Show the progress messages as the disk is dumped, restored, or copied. If PIPEDDR is called as a Pipeline stage and a secondary output stream is connected, these messages are output to that stream instead of being typed on the console.

**NOType**
Do not show the progress messages as the disk is dumped, restored, or copied. The option QUIET is accepted as a synonym for this option.

**NOCRC**
Do not add any verification code (CRC, CKSUM, or DIGEST) to the end of the output.

**CRC**
Add a 32 bit Cyclic Redundancy Code (CRC) to the end of the data dumped. The CRC value is the last record of the file. See "Usage Note 8. Ensuring data integrity with CRC, CKSUM, SDIGEST, or DIGEST" on page 22 for more information about data verification.

**CKSUM**

Compute a checksum (binary value) of the dumped disk image and append it to the end of the data. This checksum value is compatible with the value computed by the Posix cksum command. The cksum command is also available on most Linux® systems. The checksum is the last record of the file. See “Usage Note 8. Ensuring data integrity with CRC, CKSUM, SDIGEST, or DIGEST” on page 22 for more information on data verification and “Usage Note 9. Output file structure with CKSUM or DIGEST verification” on page 23 for more information when this option is specified. The file is structured so that the cksum of the disk image can be calculated and verified on a Posix or Linux system.

**DIGest**
**SDIGest**

Compute a message digest (binary value) of the dumped or transferred disk image and append it to the end of the data. This allows the integrity of the disk image to be verified when the disk is restored and may allow you to verify the backup file itself. See “Usage Note 8. Ensuring data integrity with CRC, CKSUM, SDIGEST, or DIGEST” on page 22 for more information on the data verification of a disk image. The SDIGEST (single digest) option is enabled by default for disks transferred over TCP/IP. The DIGEST option adds 2 digest values to the output file. See “Usage Note 9. Output file structure with CKSUM or DIGEST verification” on page 23 for more information. The file is structured so that the digest of the disk image can be calculated and verified on a Posix or Linux system.

These digest types are supported:

**SHA256**

Compute a 32-byte message digest according to FIPS 180-2. If no digest format is specified, this is the default. The message security assist feature is used if it is available on the hardware.

**MD5**

Compute a 16-byte message digest according to RFC 1321.

**SHA1**

Compute a 20-byte message digest according to RFC 3174. The message security assist feature is used if it is available on the hardware.

**SHA384**

Compute a 48-byte message digest according to FIPS 180-2.

**SHA512**

Compute a 64-byte message digest according to FIPS 180-2. The message security assist feature is used if it is available on the hardware.

**NOCIpher**

Do not encrypt the output.

**CIpher**

Encrypt the output using the CPACF hardware feature. The argument specifies the cipher method to use. See “Usage Note 10. More about the encryption options” on page 23 for more information and information about any additional packages you may need.

These cipher formats are supported:

**AES**

Use the Advanced Encryption Standard algorithm with a key length of 128, 192, or 256 bits. The key must be 16, 24, or 32 bytes respectively.

**TDES**

Use the triple Data Encryption Standard algorithm. The key must be 16 or 24 bytes.

**DES**

Use the Data Encryption Standard algorithm when an 8 byte key is supplied. If a longer key is needed, use triple DES.

**KEY** *keyvalue*

Specify the cipher key to use. The KEY option is ignored if the Cipher option is not specified. The keyvalue must be 16, 24, or 32 bytes for the AES method, 8, 16, or 24 bytes for the DES method, and 16 or 24 bytes for the TDES method. The keyvalue can be a single word following the keyword KEY, a hexadecimal string (2 hex digits for each byte of the key), or a character string enclosed in delimiters. The valid delimiters are quotes (single or double) or parentheses. This allows spaces to be specified as part of the key. The same delimiter must be used at the beginning and end of the key string. (A closing parenthesis is used to end an opening parenthesis.) Mixed case keys are supported. Note that if the key itself begins with a quote or parenthesis, then one of the other delimiters must be used. If the key is a single word that consists of only valid hexadecimal characters, it must be specified as a hexadecimal string or enclosed in delimiters. For instance, an 8 byte key of `FeeD1caB` is specified as either `KEY C68585C4F18381C2` or `KEY(FeeD1caB)`. A key value is required if the CIPHER option is specified. It can be supplied using this option, the **KEYVARIABLE** option, or stored in GLOBALV using the SET command. See "Usage Note 10. More about the encryption options" on page 23 for information about storing a key.

**KEYVARiable** *variablename*

If PIPEDDR is being invoked from another Rexx EXEC, specify the name of a variable in that program that contains the cipher key. If PIPEDDR is invoked from the command line, or the specified variable is unset or is empty, then a message is issued and the key must be supplied by one of the other methods. This helps suppress the key from appearing in trace output or console logs. If a key is obtained from the variable, it has priority over any other method of specifying the key.

**NOCBC**

Do not use default Cipher Block Chaining mode. If the data is encrypted, Electronic Code Book (EBC) mode is used. Note that NOCBC mode was the default in older versions of PIPEDDR, so the NOCBC option may be required to decrypt a file created with an older version.

**CBC**

Specify that Cipher Block Chaining (CBC) mode is used for encryption or decryption. This is the default encryption mode. Also see "Usage Note 10. More about the encryption options" on page 23.

**NOPRompt**

On a Restore command, do not prompt for a Yes or No response to confirm output disk user ID and address. This option is ignored on a Dump command.

**MATCHDisk**

On a Restore command, check that the output disk is restored to the same user ID and virtual address that it was dumped from. Note that the disk does not need to be restored on the same system. If the user ID and virtual address do not match, an error is displayed and the disk is not restored. This option is ignored on a Dump command.

**ONEread**

Normally for a restore from disk, tape, ftp, or http, the exec reads the input file to obtain the disk metadata information (the first line of the file), processes that data, then reads the input file again from the beginning to restore the disk. If this option is specified, the input file is only read once. This option is implied if the **FROM FILEDEF** or **FROM PIPE** methods are specified. Using this option may avoid problems using tape or remote servers, but less error checking of the input data format is done. It is recommended that this option only be used if errors are encountered. This option is ignored on a Dump command.

When this option is used or implied, you must supply additional command options that describe how the disk was dumped originally. These include encryption options, how the data was compacted, and

if a verification value is included. Otherwise, it is likely that error messages will be shown when the exec attempts to restore the data.

**NOSIZECheck**
**NOSIZEChk**
Do not check that the original input disk and the specified restore disk are the same size. This option is ignored on a Dump command. Normally, the input and output disk sizes are checked to verify that they match, and if they do not match, error messages are issued and the output disk is not changed. If this option is specified, the check is still performed and messages are issued, but the disk is restored anyway. Other error messages may be displayed if PIPEDDR attempts to write beyond the end of the output disk. See "Usage Note 7. Restoring data to a disk of a different size than the original disk" on page 22 for additional information about using this option.

**TRUNCate** *n*
Only dump a partial number of cylinders or blocks. This option is ignored on a Restore command. The value *n* specifies the last cylinder or block of the disk that is dumped to the file or sent to a remote system. Note that the numbering of cylinders or blocks on a disk begins with zero, so the total number of cylinders or blocks that are dumped is one more than the truncate value. The data must be restored to a disk with the same number of cylinders or blocks that were dumped, which means the size of the disk must be one more than the truncate value. Or, the NOSIZECHECK option must be specified on the restore command to restore to a disk of a different size. Be sure you understand the format of the partial data you are dumping, otherwise the restored disk may not be usable.

**STAble**
On a **Dump** command, attempt to use the "SR" (stable read) link option to access the source disk. If the user is not allowed to use this link mode, then use link mode "R", which will fail if any other user has a R/W link to the disk. On a **Restore** command, attempt to use the "EW" (exclusive write) link option to access the target disk. If the user is not allowed to use this link mode, then use link mode "W" instead, which will fail if any other user has a link to the disk. On a **Copy** command, use the modes described above to link both the source and target disks. The intent of this option is to prevent dumping a disk another user may be updating or want to update and to prevent restoring to a disk another user may be reading or wants to read before it is restored. For any command, if a disk is already linked by the user, this option does not attempt to re-link the disk. The existing link is used.

**RAW**
On a **Dump** command, create an output file with no header or footer data added to the disk image. Normally, PIPEDDR adds a header line to the output file to document the size, disk type, and which options were specified to dump the input disk. The CRC, CKSUM, or DIGEST options add a footer record with a CRC or digest value. If the RAW option is specified, no header is added and the CRC, CKSUM, or DIGEST options are not valid. The data can be compressed if one of those options are specified. The CIPHER option cannot be specified with this option. On a **Restore** command, the same options used to dump a disk with RAW must also be specified. See "Usage Note 7. Restoring data to a disk of a different size than the original disk" on page 22 for additional information about using this option.

**DEBUG**
Show the version of the PIPEDDR EXEC. If no function is specified, just show the version and exit. For example, entering `PIPEDDR (DEBUG` shows the version and exits with return code 1. If a function is entered and ftp or http parameters are specified for the transfer, display additional information about the transfer. For ftp, the trace option is passed to the ftp stage. For http, the sent and received header information is displayed.

## Remote Options

These options apply to remote connections.

**SECure**
Use a secure (SSL/TLS) socket for the connection used to perform a remote dump or restore. If the VM SSL server is not configured on both the sending and receiving system, the connection will fail.

**TLSLabel** *label*
Specify the label of the certificate defined in the VM system SSL certificate database that is used to secure the connection. The TLSLABEL option implies the SECURE option. A TLS label can only be specified the server (receiving) side of a remote transfer, therefore this option is only valid with the FROM REMOTE keywords on a **Restore** command or with the deprecated LISTEN option. A label must be 1 to 8 alphanumeric characters.

**UNSAFE**
Add the keyword UNSAFE to the secure tcpclient connection to the receiving system. This option is only valid if one of the SECURE or TLSLABEL options are also specified on a DUMP command to a remote system. For secure connections, normally tcpclient verifies that the destination name matches the identity of the server stated in the server certificate. This only done when the destination is specified as host name or host name with domain. To skip this verification, specify UNSAFE.

**TIMEOut** *seconds*
This option is only valid with the FROM REMOTE keywords on a Restore command or with the deprecated LISTEN option. It specifies the number of seconds PIPEDDR will wait for an incoming connection. After the specified number of seconds, if no incoming connection is established, the exec ends with an error and a non-zero return code.

**POrt** *portnum*
Specify the TCP/IP port used to listen for a connection from a remote system or the listening port number on the remote system where PIPEDDR is waiting. The PORT option is optional on a **Restore** command but required on a **Dump** command. The TCP/IP port can also be supplied as a positional argument to PIPEDDR, but that value is ignored if the PORT option is also specified.

## FTP and HTTP Options

These options apply to FTP and HTTP selections.

**PASSWORDVARiable** *variablename*
If PIPEDDR is being invoked from another Rexx exec, specify the name of a variable in that program that contains the ftp password. If PIPEDDR is invoked from the command line, or the specified variable is unset or is empty, then a message is issued and PIPEDDR will prompt for the ftp password. This helps suppress the password from appearing in trace output or console logs. If a password is specified in the url or in the ftp command options, it is used instead of the password in this variable.

**EPASSIve**
**EPSV**
If the input or output file is on an FTP server, and the FTP REXX stage is available and used, add the EPASSIVE option to the arguments to that stage. The FTP stage will attempt to establish the connection using Extended Passive mode. EPSV is a synonym for EPASSIVE. This option only applies when an FTP source or destination is used and only for the FTP REXX stage. See “Usage Note 2. FTP support in Pipelines” on page 19 for more information.

**PASSIve**
**PASV**
If the input or output file is on an FTP server, and the FTP REXX stage is available and used, add the PASSIVE option to the arguments to that stage. The FTP stage will attempt to establish the connection using Passive mode. PASV is a synonym for PASSIVE. This option only applies when an FTP source or destination is used and only for the FTP REXX stage. See "Usage Note 2. FTP support in Pipelines" on page 19 for more information.

**NOPASSIve**
**NOPASV**
**NPASV**
If the input or output file is on an FTP server, and the FTP REXX stage is available and used, allow that stage to use its default connection mode, which uses the FTP PORT command. NOPASV and NPASV are synonyms for the NOPASSIVE option. This option only applies when an FTP source or destination is used and only for the FTP REXX stage. See "Usage Note 2. FTP support in Pipelines" on page 19 for more information.

**VMFTP**
Specify this option if the receiving FTP server is hosted by z/VM. This option adds the correct "site" option to the ftp stage so that the disk image file is created with the correct record format.

**FTPSITE** *siteargs*
Specifies an optional SITE command that is issued by the FTP client. The command is issued before data transfer, after navigating to the correct directory. It is not translated to upper case. If the SITE command is multiple words, enclose the command in delimters. The valid delimters are quotes (single or double) or parentheses. The same delimiter must be used at the beginning and end of the string. A closing parenthesis is used to end an opening parenthesis. For example: `FTPSITE (umask 022)`

**UNSAFE**
Add the keyword UNSAFE to the secure ftp connection to the server. This option is only valid for an "ftps" connection. When the connection is made, the ftp pipelines stage verifies that the destination name matches the identity of the server stated in the server certificate. This only done when the destination is specified as host name or host name with domain. To skip this verification, specify UNSAFE.

**OLD**
Force the exec to use an old option or method. See "Usage Note 2. FTP support in Pipelines" on page 19 for a description of the 3 different ftp stages that PIPEDDR may use. If Pipelines supports the built in FTP stage, it is always used and this option is ignored. If the FTP REXX pipeline stage is available, it is used unless the OLD option is specified. This option forces the exec to use the alternate ftpget and ftpput FTP pipeline stages instead of FTP REXX.

**POrt** *portnum*
Specify the TCP/IP port on the server used for the connection. If a port is specified as part of the url, this option is ignored. A port specification in the URL is the preferred way to specify the port.

## Deprecated Options

These options remain functional but the Target arguments are recommended.

**FILEdef**
Dump or restore a disk to a CMS Filedef. The filedef must be defined with the CMS FILEDEF command before PIPEDDR is invoked. Any type of filedef can be defined that is usable on the Pipelines "qsam" stage. Any error messages shown when using this method are from CMS Pipelines. The default output record format is Fixed 1024 byte records. The ONEREAD option is implied when using this option with a Restore command.

**LIsten**
On a Restore command, wait for an incoming TCP/IP connection from a remote system running a PIPEDDR Dump TO REMOTE command. If a TCP/IP port is not specified using the PORT option or with the positional arguments to PIPEDDR, the next available port is obtained and the listening port number is displayed on the console. PIPEDDR will keep waiting for a connection until one is established or the pipeline is stopped.

**FTP**
Dump to or restore from a file on an ftp server. If an FTP URL is specified (ftp://) as the option, then the FTP option is implied. See "Usage Note 4. FTP and HTTP options" on page 20 for more about this option.

**HTTP**
Restore from a file located on an http server. This option is invalid on a Dump command. If an HTTP URL is specified, the HTTP option is implied. The file may be placed on the server via ftp, a CMS file of the dumped disk was transferred to the server in binary format, or the CMS file is hosted by a HTTP server running on CMS.

*url* is either a list of options to the ftp pipeline stage (which can also be used for an HTTP transfer, except the -userid and -password options are ignored), or a normal web URL. It must be enclosed in parentheses if a URL is not used or other options follow. See "Usage Note 4. FTP and HTTP options" on page 20 for more information about specifying a URL.

**TAPE**
Dump a disk to tape or restore a disk from tape. A tape drive virtual address can be specified after the TAPE keyword. If no address is specified, it is assumed the tape drive is attached at virtual address 181.

**BFSPath** *path*
Specify the path in the BFS filesystem where the input or output file resides. The path must already be mounted before running PIPEDDR. The synonym HFSPath can be specified instead of BFSPath. The BFS option is implied when the BFSPATH option is specified.

**BFS**
Specify that the input or output file resides in BFS, but no path is specified. The file will be read from or written to the current working directory set via the OPENVM SET DIRECTORY command. See the description of the BFSPath option for more information. The synonym HFS can be specified instead of BFS.

# Usage Notes:

## Usage Note 1. Disk input/output formats

The PIPEDDR EXEC supports 2 different methods for directly reading from and writing to disks. The first method uses PIPE's TRACKREAD and TRACKWRITE stages for ECKD disks and the FBAREAD and FBAWRITE stages for FB-512 disks. This is the default input and output method. z/VM 6.4 and later already have the necessary levels of these stages. z/VM 6.4 with APAR VM66139 (and later) also support extended address volumes, which are disks larger than 65520 cylinders.

In z/VM 6.3 and earlier, these stages are not included with the level of Pipelines that is available by default on z/VM. They require the *Runtime Distribution* level of Pipelines available from *http://vm.marist.edu/%7Epipeline/index.html#Runtime* at version 110B0004 (15 May 2002 level) or later for reading and writing ECKD disks (level 110C000A (24 Jul 2012 level) for ECKD disks larger than 32767 cylinders) and level 110C0004 (01 Jul 2010 level) or later for reading and writing FBA disks.

The second method is used for dump and restore if the CMSDDR or VMDDR option is specified or defaulted; or if the exec is renamed to CMSDDR or VMDDR (or invoked using a synonym of either CMSDDR or VMDDR.) This method uses the Pipelines interface to DDR available with z/VM 6.2 and later. This interface diverts records that would normally be written to tape to the pipeline. The control records that are written include the current time of day clock. Dumping the same disk image multiple times will result in files with slightly different contents. Older releases can obtain the modified DDR module and DDR pipe stage from the DRPC package on the VM download page. (See *http://www.vm.ibm.com/download/packages/descript.cgi?drpc*). Both the DRPC MODULE and the correct DDR MODULE must be available when the CMSDDR option is specified. Unless you are on z/VM 6.2 or later, the DDR MODULE on the S disk of your z/VM system will not work.

## Usage Note 2. FTP support in Pipelines

PIPEDDR can use 3 different ftp stages. The newest ftp stage is built in to Pipelines and is available if secure TCP/IP connections are supported by CMS Pipelines. (APAR VM66365 for z/VM 7.1, included in z/VM 7.2 and later.) This stage supports both secure and unsecured ftp sessions. The other ftp stages supported by PIPEDDR are not built-in to Pipelines; they are loaded from a separate module or file. If the built in stage is not available, the preferred external ftp stage is the FTP REXX stage downloaded from the VM download page. The other external stage is an ftp stage supplied with z/VM that are part of the install process. Neither of these stages support secure ftp sessions.

The FTP REXX stage can be found in the *FTPREXX package* on the VM downloads page. See *http://www.vm.ibm.com/download/packages/descript.cgi?ftprexx* to obtain the package. If FTP REXX is available, you can still force PIPEDDR to use the other ftp stages (described in the next paragraph) by adding the OLD option. The source code for the FTP REXX stage is provided, although still a tool without any formal support.

The stages supplied with the z/VM install process are contained in either the DRPC MODULE which found on the S (190) disk or from the INSTPIPE MODULE (normally found on either MAINT*vrm* 222, 4CC, or MAINT 193). The DRPC module is also part of the *DRPC package*. on the VM download page, if your level of VM does not have it. See *http://www.vm.ibm.com/download/packages/descript.cgi?drpc*. If the ftp stage is built into Pipelines, it is used. Otherwise, FTP REXX is used if it is found. If it is not found or the OLD option is specified, then if the ftp stages are not already loaded before the exec is started, PIPEDDR will look for DRPC MODULE first and then INSTPIPE MODULE if DRPC is not found. If the ftp stages can't be found, the exec ends with an error. These installation FTP stages are not part of PIPEDDR, are not officially supported by IBM, and no source code is provided.

## Usage Note 3. More about the URL argument

A url is used to refer to a file on a remote system using a standard convention. PIPEDDR supports reading from an HTTP server and reading and writing to an FTP server if an FTP pipeline stage is available. Secure connections (HTTPS, FTPS) are only supported if your level of Pipelines supports secure connections. SFTP connections are not supported.

A basic URL or URI has this structure:

<*scheme*>://<*user*>:<*password*>@<*host*>:<*port*>/<*url-path*>

PIPEDDR supports a <*scheme*> of HTTP, HTTPS, FTP, or FTPS. (Secure connections are only supported if Pipelines supports these connections.) The remote server is found on host <*host*>. If the <*port*> is specified in the url, then that is the port used instead of the standard port.

For FTP, the user ID <*user*> is used to authenticate to this host, using password <*password*>. The user ID must be specified, even if the “anonymous” user ID is used. In the URL, special characters

are encoded using the percent sign such as %2f to represent a forward slash ("/") character. The hexadecimal value after the percent sign is the ASCII code for the character. No user ID or password is supported for HTTP requests. If the password is not supplied and the user is not "anonymous", then PIPEDDR prompts for the password. A NETRC DATA file can be used to supply the user ID and password (see number 1 under "Notes for FTP transfers") or the **PASSWORDVARiable** option can be used to supply the password. Do not use percent encoding of special characters in the password unless the password is part of the URL.

The <*url-path*> can specify just a path to a directory or a path and a file name. If no filename is specified, the default file name of <*userid*>.DISK<*nnnn*> is used. All uppercase characters are translated to lower case and <*nnnn*> is replaced by the virtual address of the source disk.

Notes for FTP transfers

1. User IDs and passwords for ftp servers can be stored in a NETRC DATA file instead of specifying one or both of them on the command line, passing a password via a variable, or responding to password prompts. The format of the NETRC DATA file is documented in the *z/VM TCP/IP User's Guide*, "Appendix B. Using the NETRC DATA File." The format of each line is:

   **machine** *foreign_host* **login** *user_id* **password** *password*

   PIPEDDR only reads the first NETRC DATA file it finds in the search order. It finds the first match of either the hostname and user ID or just the hostname, depending on if a user ID was specified as part of the URL. The password in NETRC DATA is ignored if the password is included in the URL or supplied in a variable.

2. If a user ID of anonymous is not used, and a password is not specified in the url, is not available via a variable, or in a NETRC DATA file, the PIPEDDR exec prompts for the FTP password. The password will not appear when typed, and will not appear in any spooled console log. The password prompting that is built-in to the ftp pipeline stages is avoided. Note that to use anonymous ftp login, the user ID must be specified as "anonymous". If the user ID is not specified, the exec ends with an error.

Notes for HTTP transfers

1. If the DEBUG option is specified, the http request record and all headers returned by the server are displayed on the console. This may be useful to debug problems with http transfers.
2. The HTTP option does not use additional pipeline modules, only stages already built-in to the minimum required pipeline level are used.

## Usage Note 4. FTP and HTTP options

**Note:** These options are deprecated, the preferred specification is a url as a target argument that follows the keyword **FROM** or **TO**. Secure connections cannot be specified using these options.

The ftp arguments or an ftp url are passed to the ftp pipeline stage by the FTP option. This option can be specified as "FTP (<ftp arguments>)" with other options following it, as the last option on the command line as "FTP <ftp arguments>", or as a URL in standard form that starts with "ftp://".

HTTP arguments can be specified the same way using the keyword HTTP instead of FTP, or using a standard HTTP URL.

The FTP arguments are any parameters valid for the ftpget or ftpput pipe stages. Even if the FTP REXX pipeline stage is available, the same arguments can be used. The exception is the -debug option. Use the DEBUG option to get additional output from any ftp pipeline stage. The same arguments are valid for HTTP, but it is more common to use an HTTP URL for HTTP transfers. The ftp parameters usable for PIPEDDR are:

**-h,-host,-hostname** *hostname*
hostname or IP address to connect to

**-u,-user,-userid** *userid*
user ID to connect to - defaults to 'anonymous'

**-p,-pw,-pwd,-password** *password*
will prompt if not supplied

**-port** *nnnn*
FTP port - defaults to 21

**-d,-dir,-directory** *directory*
directory to change to before file transfer

**-f,-file,-filename** *filename*
name of the file to be written onto the FTP server

**-debug**
turns on debugging output

Some notes about these options:

- PIPEDDR always writes files sent to FTP in Fixed 1024 format, so the FTP option -fixed 1024 is automatically added to the ftp argument string for the restore command if the ftpget pipeline stage is used.
- When using FTP for the restore command, the ftp file name will be displayed by the ftpget stage 2 times on the console unless the ONEREAD option is also specified. If the ONEREAD option is specified, the input file is only read 1 time.
- If the -debug argument is specified as an ftp argument or with the DEBUG command option, the ftp user ID and password may be displayed as part of the debug output.
- If an HTTP URL is not used and ftp parameters are used instead, the only ftp arguments valid for http are -hostname, -port, -directory, and -filename. Any of the other ftp options are ignored.

## Usage Note 5. Using BFS and NFS files for input and output

PIPEDDR can read and write OpenExtensions files, which are found in the Byte File System (BFS), using the CMS Pipelines HFS stage. The most useful aspect about this support is that the CMS NFS client can be used to mount NFS filesystems on remote servers into the byte file system. This allows PIPEDDR to dump or restore disks from files on remote NFS servers. To use this support, your user ID must be allowed to mount the root BFS filesystem and you must be allowed to either write data in BFS or mount an NFS file into BFS. For example:

```
OPENVM MOUNT /../NFS:vmsys.example.com/export /bfshome (ANONYMOUS NOTRANSLATE
```

For more information, see the z/VM documentation *OpenExtensions Commands Reference* and the *OpenExtensions User's Guide*. You can find these documents on the *z/VM Library Overview* web page at *http://www.vm.ibm.com/library/*.

## Usage Note 6. Notes about compression

1. The TERSE option can only be used if the TERSE Pipelines stage is available. This allows the exec to create output files that are smaller or send less data over the network. The TERSE Pipelines stage is part of the PIPSYSF Pipelines filter package which must be downloaded from the Pipelines Runtime Distribution page. See *http://vm.marist.edu/%7Epipeline/pipsysf.html* for this package.

2. The DDR module supports LZCOMPACT compression, which uses hardware instructions to compact the data. This format is not compatible with the TERSE format, but can be used to create more compact output.
3. The ZLIB option can only be used if the ZLIB Pipelines stage is available. This stage is found in the ZLIBSTG MODULE file. You can extract this module from a VMARC format file downloaded from John Hartmann's space on GitHub. Go to *https://github.com/jphartmann/cmslib-exec* and download the file “fplgcc.vma”. Upload it to z/VM as FPLGCC VMARC in binary with file format Fixed 80. Use the command `VMARC UNPK FPLGCC VMARC A ZLIBSTG MODULE A` to obtain the module file. PIPEDDR will automatically load the module when the ZLIB option is specified or when a ZLIB compressed disk is restored. More information about zlib compression can be found at *http://zlib.net/*

## Usage Note 7. Restoring data to a disk of a different size than the original disk

If the **NOSIZECHECK** or **RAW** options are used, PIPEDDR does not check that the size of the output disk is the same size as the original input disk. Using the **RAW** option means that there is no disk information header on the file and all processing options must be specified when the disk is restored. The **NOSIZECHECK** option performs all other checking except checking that the size of the output disk is the same as the actual input disk. If the **TRUNCATE** option is used to dump the disk, the data must be restored to a disk that matches the specified truncated size unless the **NOSIZECHECK** option is specified. This allows you to restore a disk image or perform a copy on a disk that is larger or smaller than the original disk or the truncation value. Restoring to a smaller disk will lose some of the data. Restoring to a larger disk will write all of the data, but the disk may appear to only be the size of the original input disk.

## Usage Note 8. Ensuring data integrity with CRC, CKSUM, SDIGEST, or DIGEST

The intent of these options is to compute a value based on the disk image and append that value to the output file. When the file is read to restore the disk, the value is computed again and compared to the original value. If the values do not match exactly, the data in the disk image file has changed and the disk will not restore correctly. These options use the corresponding Pipelines stage (CRC or DIGEST) and more information about the algorithms used are found in the help information for those stages.

Specifying one of these options is normally not needed on a restore command because a keyword is added to the header record which automatically enables verification of the data during a restore. However, if the **ONEREAD** option is specified on a restore and the input data includes a verification value, the corresponding option must be specified so that the proper check is performed. Note that if an older level of PIPEDDR that does not support one of these options is used to restore a disk, an error message is displayed because the older levels do not recognize the keyword in the header.

Here is a short description of each type:

CRC and CKSUM - Calculate a 32 bit (4 byte) Cyclic Redundancy Code of the data. The CRC option uses the “CRC-32” calculation and the CKSUM option uses the CKSUM calculation. Also see “Usage Note 9. Output file structure with CKSUM or DIGEST verification” on page 23 for more information. The CRC option appends the CRC value to the end of the original input data, before any compression or encryption processing.

SDIGEST and DIGEST - Calculate a message digest. This is a more modern and longer value that should ensure that the data has not been altered. The result of the calculation is a binary digest value from 16 to 64 bytes long. The length depends on which digest type is selected. The SDIGEST (single digest) option appends a single digest value to the file in the same way a CRC value is added as described above. The DIGEST option adds 2 digest values to the output file. The first is the single

digest value that was just described. The second is computed after any compression and encryption is performed. Both of these are recalculated and checked when the disk is restored. Also see “Usage Note 9. Output file structure with CKSUM or DIGEST verification” on page 23 for more information about DIGEST.

If the **SDIGEST** or **DIGEST** options are specified without selecting a type, the default SHA256 digest is computed which produces a 32 byte result. This default was chosen because it uses hardware instructions when the machine feature is installed. The DIGEST stage does not support hardware calculation of SHA384 and SHA512 message digests. The choices, along with their digest lengths (in bytes) are: MD5 (16), SHA1 (20), SHA256 (32), SHA384 (48), and SHA512 (64). For more information see *http://enwp.org/SHA-1* and *http://enwp.org/SHA-2*. Also see “Usage Note 9. Output file structure with CKSUM or DIGEST verification” for more information about verifying the data using the digest value. “Usage Note 11. More about incrementally updating a remote disk” on page 24 describes how a digest value is used to update a remote disk.

## Usage Note 9. Output file structure with CKSUM or DIGEST verification

The intent of the **CKSUM** or **DIGEST** options is to allow the verification of the checksum or message digest of the disk image when the file is stored on other platforms. If the **CKSUM** option is specified, the checksum of the disk image can be verified using the Posix cksum command. If one of the **DIGEST** methods is selected, many platforms have the necessary commands to calculate the message digest value of the disk image. For example, a disk image sent to a Linux ftp server by PIPEDDR can be validated on that server that it is unchanged.

The output file created by PIPEDDR when one of these options is specified has a 1024 byte header that contains metadata about the disk image that was dumped and a 1024 byte footer with the validation values. The remaining data between the header and the footer is the actual contents of the disk after compressing and/or encrypting the disk. If the file is stored in the CMS filesystem, it remains a fixed length file with 1024 byte records.

The header contains metadata about the dumped disk in both EBCDIC and ASCII with a hex x'0A' character before and after the ASCII formatted data. There are binary zeros in the rest of the record. If the eighth blank delimited word of the header is “CKSUM” then the verification record is in checksum format. In this format, the footer contains the binary cksum in the first 4 bytes and the length of the input file in the next 8 bytes. The length of a checksum is 12 bytes, indicated by a 12 in the twelfth word of the header. If the eighth word of the header is “DDIGEST” (or “DIGEST”) then the eleventh word indicates the digest format and the twelfth word indicates the length of the digest in bytes. After the checksum or digest value in the footer, a readable hexadecimal version of the same value is translated to ASCII and placed in the record. There are binary zeros in the rest of the record. For example, the length of the default SHA256 digest is 32 bytes, so the first 32 bytes of the final 1024 byte record is the digest in binary. The next 64 bytes are the ASCII translation of the digest value. Binary zeros fill in the rest of the record.

More information about how to use the cksum or digest information to verify the contents of a file can be found in the section “How to verify a file” on page 26.

## Usage Note 10. More about the encryption options

To encrypt or decrypt data, your system must have the “CPACF” feature enabled. The KM package contains CSL routines that interface to the hardware instructions that actually do the encryption and decryption. Go to *http://www.vm.ibm.com/download/packages/descript.cgi?KM* and download the VMARC archive to obtain the CSL routine. PIPEDDR only requires the KMRTNS CSLLIB file from the package.

The default encryption mode of operation (not the encryption type) is CBC, Cipher Block Chaining mode. The encryption of each cipher block depends on the encryption of the previous block. A random initial block, called the Initialization Vector, is used to ensure that encrypting the same input data again will result in different output (ciphertext.) If the NOCBC option is specified, EBC (Electronic Code Book) mode is used. Each block of a message is encrypted separately and the output will be the same each time the same input data is encrypted. See *http://enwp.org/Block_cipher_mode_of_operation* for more information.

If a cipher stage is available in CMS Pipelines, it will be used to for symmetric encryption or decryption of data. Data encrypted by the Pipelines stage can be decrypted by the KM CSL routines and vice versa. Both methods use the same hardware instructions.

Previous versions of PIPEDDR defaulted to the EBC encryption method. The default is now the CBC method. To decrypt a file created by a previous version, you may need to specify the NOCBC option to force it to use ECB mode for the decryption.

There is no method available in CMS to securely store an encrypted or secure key, so be aware that the key will appear in memory and possibly on the console and/or on disk in clear text. PIPEDDR allows you to pass the key via a variable in the calling program or store the key in GLOBALV using the PIPEDDR SET KEY command. The key is stored as a hexadecimal string. To view the stored key, enter PIPEDDR SET QUERY EXT. You may store the key in temporary (in memory only) GLOBALV by any means you choose so that the key does not appear on the command line and is not stored on disk. Make sure the key you create is the correct length for the encryption method you select. A Rexx example:

```
keyvalue='this IS my key!!'
'GLOBALV SELECT $PIPEDDR PUT KEYVALUE'
```

Or, using the KEYVARIABLE option:

```
/* Call a routine that returns a password protected key */
secretkey=GetKey()
'EXEC PIPEDDR DUMP MAINT 190 (CIPHER AES KEYVARIABLE SECRETKEY'
```

## Usage Note 11. More about incrementally updating a remote disk

The TO REMOTE function of PIPEDDR sends the complete contents of a disk to a remote system over the TCP/IP network. The disk is sent track by track or a set of blocks at a time. If the disk was copied at an earlier time to the remote system and has not had extensive changes, the TO UPDATE function can be used to only update the changed tracks or blocks. When PIPEDDR operates this way, the remote system reads part of the existing disk and generates a hash value (using the Pipelines digest stage) of each track or set of blocks and sends that value to the sending system. The sending system also reads the same part of the disk and generates the same type of hash value from it. If the values are different, then the track or blocks are sent to the remote system and written to the disk. Only changed tracks or blocks are sent. The encryption options are supported, so the disk data that changed can be encrypted before it is sent to the remote system.

Use this method if you believe that only a limited number of changes have been made to the disk. If a large number of changes have been made, the entire disk should be sent with one of the compression options enabled.

This function was inspired by Rob van der Heij's *Sir Rob the Plumber* blog entries on remote disk copies. Please see the first blog entry at *https://rvdheij.wordpress.com/2019/03/17/disk-copy-introduction/* for the introduction and the later entries on the implementation. The ideas and examples were adapted to work with the existing PIPEDDR functions.

## Errors:

PIPEDDR issues messages with message numbers and a suffix character indicating the type of message. The format of the message number is PDDRMS*nnnt* where *nnn* is the message number and *t* is the type. The type is similar to z/VM message types of E for error, W for warning, I for informational, R for response, and S for severe.

## Some examples of how to use PIPEDDR

To dump a minidisk to a file with the default name:

```
PIPEDDR DUMP MAINT 19E
```

This creates a file named MAINT DISK019E A

To dump a minidisk to a file and terse the output:

```
PIPEDDR DUMP MAINT 19E (TERSE
```

Another way is to set the default compression format to TERSE and then dump the disk:

```
PIPEDDR SET COMPACT TERSE
PIPEDDR DUMP MAINT 19E
```

To dump a minidisk to a file including the CRC:

```
PIPEDDR DUMP MAINT 19E (CRC
```

To dump the same disk to a different filemode:

```
PIPEDDR DUMP MAINT 19E TO FILE = = E
```

To dump a minidisk to a file in CMS DDR format:

```
PIPEDDR DUMP MAINT 19E (CMSDDR
```

To restore the file to the same disk:

```
PIPEDDR RESTORE MAINT 19E
```

To restore the file to a different disk and skip the prompt:

```
PIPEDDR RESTORE CMSUSER 191 FROM FILE MAINT DISK019E A (NOPROMPT
```

To dump a minidisk to a file and encrypt the data:

```
PIPEDDR DUMP MAINT 19E (CIPHER AES KEY(Super Secret KEY)
```

To restore it:

```
PIPEDDR RESTORE MAINT 19E (CIPHER AES KEY(Super Secret KEY)
```

To send an entire minidisk over the network On the receiving node, enter:

```
PIPEDDR RESTORE MAINT 19E FROM REMOTE
```

This will display the port number it is using.  To force the port to 12345:

```
PIPEDDR RESTORE MAINT 19E FROM REMOTE (PORT 12345
```

On the sending system use (where pppp is the listening port):

```
PIPEDDR DUMP MAINT 19E TO REMOTE nodeid.example.com:pppp
```

To just update a disk that was copied previouslly:

```
PIPEDDR DUMP MAINT 19E TO UPDATE nodeid.example.com:pppp
```

The connection should be made and the disk sent over.

To send a minidisk to a file named maint.disk019e on an ftp server:

```
PIPEDDR DUMP MAINT 19E TO ftp://hayden:password@server.example.com
```

To restore a minidisk from file disk.dump on an ftp server:

```
PIPEDDR RESTORE MAINT 19E FROM ftp://hayden:password@server.example.com/disks/disk.dump
```

To do the same thing but from an http server:

```
PIPEDDR RESTORE MAINT 19E FROM http://server.example.com/disks/disk.dump
```

To dump a minidisk to a file named maint.disk019e on an NFS server, first the nfs directory must be mounted to your virtual machine using OPENVM MOUNT. Assuming it is mounted at /home/maint/mnt, the command is:

```
PIPEDDR DUMP MAINT 19E TO BFSPATH /home/maint/mnt
```

---

## How to verify a file

The CKSUM and DIGEST options are designed to allow a disk image file stored on a Posix compatible platform (such as Linux) can be validated (checked that no change has occurred) on that platform.

The file created by PIPEDDR has a 1024 byte header and a 1024 byte trailer before and after the actual disk image. The header has data in blank delimited words about the input disk in both EBCDIC and ASCII, with an X'0A' byte in between the EBCDIC and ASCII translations. The ASCII translation is terminated with an X'0A' byte. The remaining data of the 1024 byte header is binary zeros.

The trailer contains the verification data which is either the cksum value or the digest value. A cksum value is always 12 bytes, but a digest value has different lengths for each type of digest. The length of the binary digest or cksum value is the twelfth word of the header. Following the binary digest or cksum value is the same value translated to ASCII text. The length of this value is twice the binary length. The remaining data of the 1024 byte trailer is binary zeros.

A simple script, such as this example, shows the ASCII header:

```
#!/bin/bash
echo "Show header (ascii version)"
# It is in the 1024 byte header, the second "line" found in there
head -c +1024 $1 | cat | head -n 2 | tail -n 1
```

The digest or cksum of the disk image part of the file can be calculated with one of these scripts:

```
#!/bin/bash
echo "Show cksum of disk contents (decimal cksum and byte count)"
# Take the 1024 byte header and trailer off and input to cksum
tail -c +1025 $1 | head -c -1024 | cksum
```

```
#!/bin/bash
echo "Show sha256 digest of disk contents"
# Take the 1024 byte header and trailer off and input to sha256
tail -c +1025 $1 | head -c -1024 | sha256sum
```

These examples can be combined into a single script that verifies the disk image has not changed. An example script named `pipeddr-verifydump.sh` is included in the same zip file as the documentation PDF file. The example script that is shown here is similar to the example included in the zip file. If you attempt to copy and paste this example, be aware that some characters may not be translated correctly.

```
#!/bin/bash
# Verify that a disk image created by PIPEDDR has not been altered.
file=$1
fileheader=$(head -c +1024 $file | cat | head -n 2 | tail -n 1)
if [ -z "$fileheader" ]; then
   echo "File $file is not in the correct format."
   exit
fi
echo "Header: $fileheader"
# Break file header into positional parameters
set $fileheader
len=${12}
hostdigest=$(tail -c 1024 $file | tail -c +$(($len+1)) | head -c $((len*2)))
if [ "$8" = "CKSUM" ]; then
   verifypgm="cksum"
#    Convert the hexadecimal CRC-32 value and length to decimal
   hostdigest="$((16#${hostdigest:0:8})) $((16#${hostdigest:9:16}))"
else
   verifypgm=$(echo ${11} | tr [:upper:] [:lower:])sum
#    Add filename "  -" to end to match command output
   hostdigest="$hostdigest  -"
fi
echo "Computing $verifypgm of the disk contents"
# Take the 1024 byte header and trailer off to compute digest or cksum
localdigest=$(tail -c +1025 $file | head -c -1024 | $verifypgm)
if [ "$hostdigest" != "$localdigest" ]; then
   echo $verifypgm "verification failed."
   echo "Value from PIPEDDR is:" $hostdigest
   echo "Local value is:" $localdigest
else
   echo "Digest values match"
fi
exit
```

These examples should help you create a more complete program to verify disk images that reside on a Posix compatible system.

## Contact:

Please send any comments or problems to the author:
Bruce Hayden, email: bjhayden@us.ibm.com

Change log of recent changes, latest changes first

```
Version    Change Description
-------    ------------------
V1.7.13    Add some options to help testing.
V1.7.12    Handle encoded urls, encode password for ftp stage.
V1.7.11    Add FTPSITE option to send FTP site command
V1.7.10    Allow delimiters of ' or " for cipher key string, fix 3380.
V1.7.9     Check that a dump is the first Pipe stage or restore has no output.
           Allow secondary output for PIPE for progress messages.
V1.7.8     Add option TRUNCATE nnn to only dump or copy part of a disk.
```

```
V1.7.7     Ignore SECURE option instead of exiting, show level with DEBUG.
V1.7.6     Add special first option DEFAULTS to ignore all preset options.
V1.7.5     Add SDIGEST (single digest) option which works like crc.
V1.7.4     Allow exec to be called as a Pipeline stage.
V1.7.3     Support new ftp stage which supports ftps.
           Add VMFTP option, to send proper SITE command for VM FTP.
V1.7.2     Support secure SSL/TLS sockets for remote dump/restore.
V1.7.1     Fix port number in error message and some spelling.
           Validate some of the fields in the file header.
           Ensure valid values for crc or digest check.
           If no input with oneread, issue error.
           RAW option supersedes ONEREAD option.
V1.7.0     Don't restrict NOPACK, and don't force it to be blocked.
           Show fewer progress messages with large disks.
           Add digest or cksum in ascii after binary value in last record.
           For remote receive or update, verify cipher key.
```