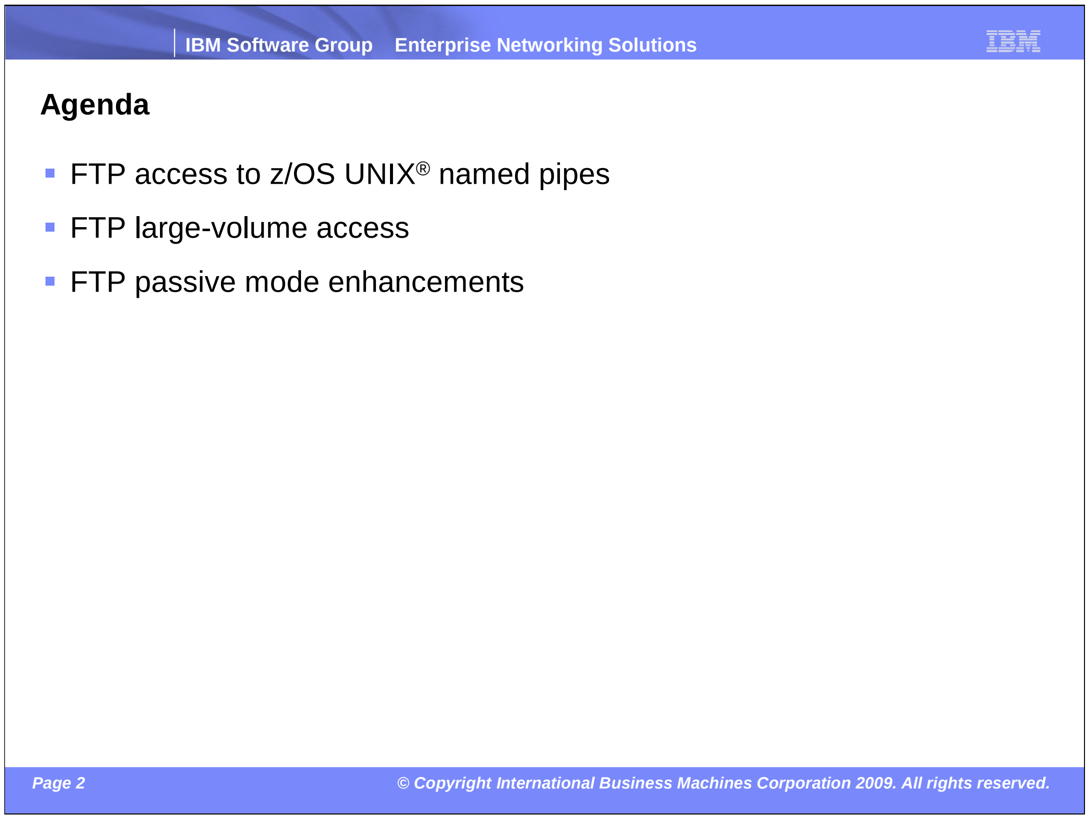
IBM Software Group Enterprise Networking Solutions
z/OS® V1R11 Communications Server

FTP enhancements

z/OS Communications Server Development, Raleigh, North Carolina

This presentation will give you an overview of the enhancements to the Communications Server in z/OS V1R11 for FTP.



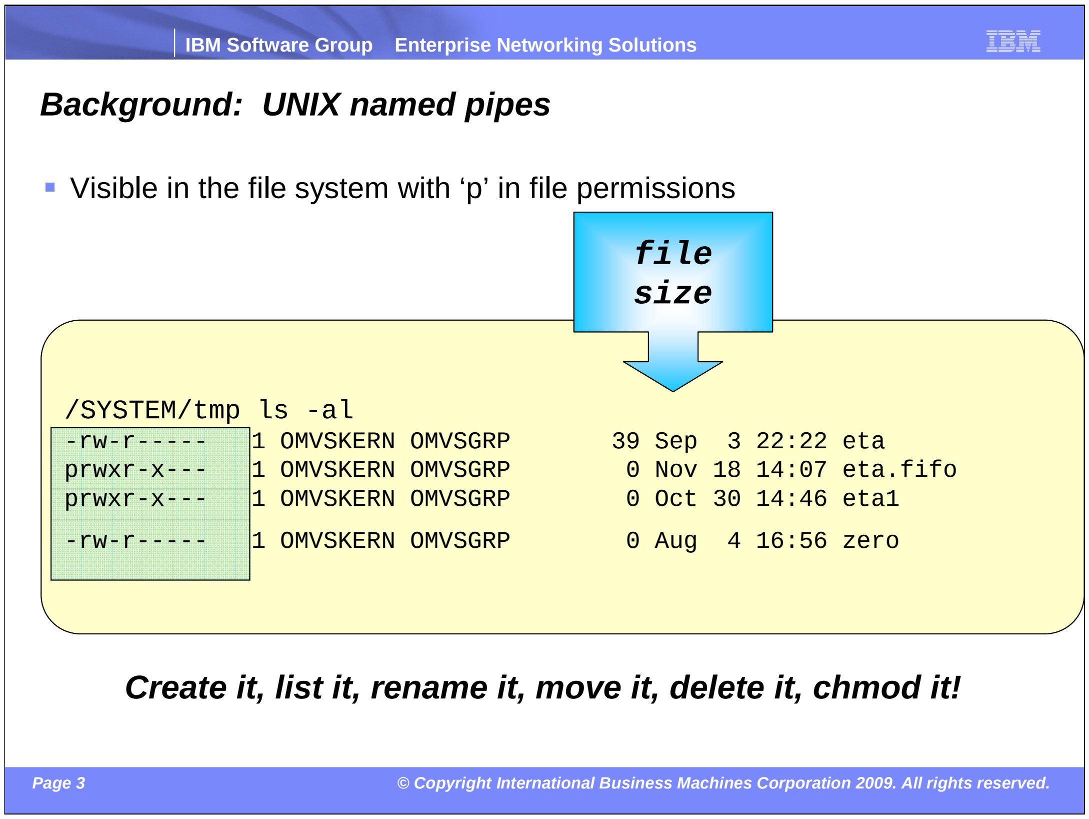
Agenda

- FTP access to z/OS UNIX® named pipes
- FTP large-volume access
- FTP passive mode enhancements

In z/OS V1R11, FTP reads and writes files to and from z/OS UNIX named pipes.

FTP reads and writes extended format sequential data sets that have been allocated in the extended addressing space (EAS) of an extended address volume (EAV). This also includes support for data sets which are eligible for such an allocation (Format 8 DSCB).

The FTP client implements an option to allow the FTP client to simulate the use of extended passive mode, even in cases where the remote FTP server does not support EPSV.



Background: UNIX named pipes

- Visible in the file system with ‘p’ in file permissions

```
/SYSTEM/tmp ls -al
-rw-r-----   1 OMVSKERN OMVSGRP       39 Sep  3 22:22 eta
prwxr-x---   1 OMVSKERN OMVSGRP        0 Nov 18 14:07 eta.fifo
prwxr-x---   1 OMVSKERN OMVSGRP        0 Oct 30 14:46 eta1

-rw-r-----   1 OMVSKERN OMVSGRP        0 Aug  4 16:56 zero
```

Create it, list it, rename it, move it, delete it, chmod it!

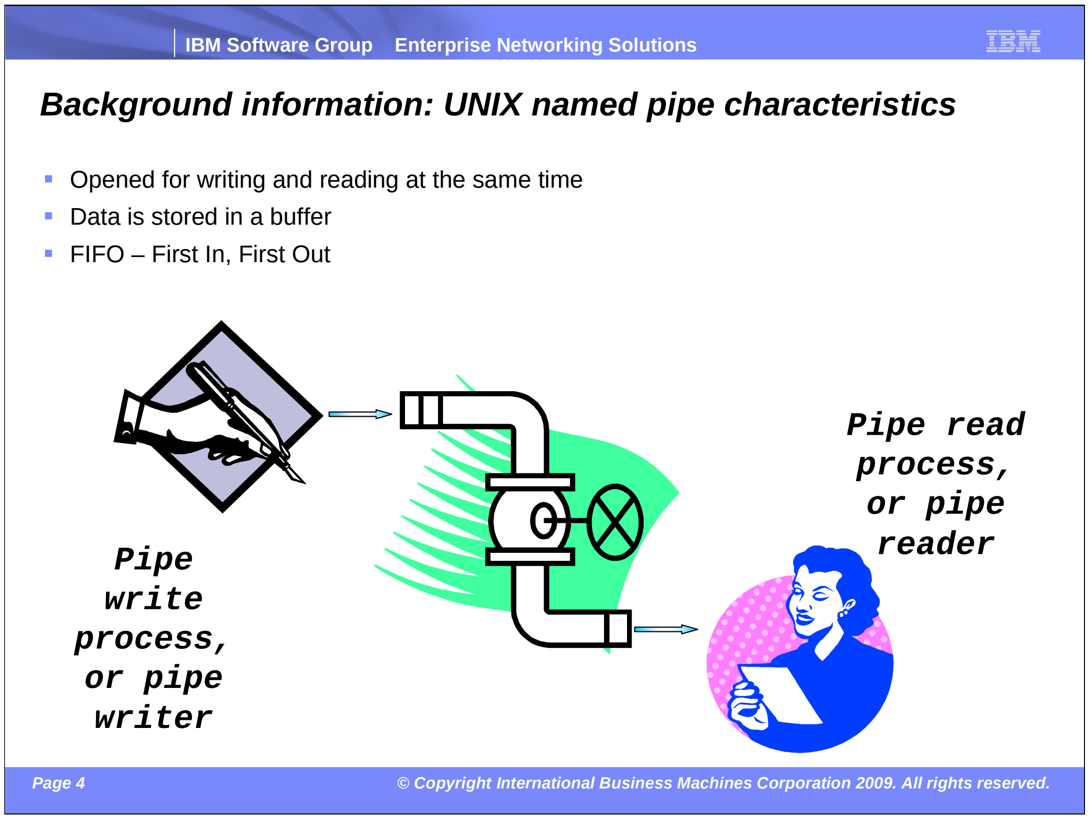
A UNIX named pipe is a UNIX pipe identified by a pathname in the file system. It has characteristics of UNIX pipes and of UNIX files. Having a pathname, a named pipe is visible in the z/OS UNIX file system. The example on this slide is the output of the z/OS UNIX **ls** command. (The output has been edited to fit on this slide.) The files listed are a combination of z/OS UNIX named pipes and z/OS UNIX regular files.

When the first character of the permissions is ‘p’ as shown for the files eta.fifo and eta1, the file is the pathname of a named pipe. The other files, 'eta‘ and ‘zero’, are regular z/OS UNIX files.

The size of the named pipes is zero. This will always be the case, even when processes are actively writing to the named pipe.

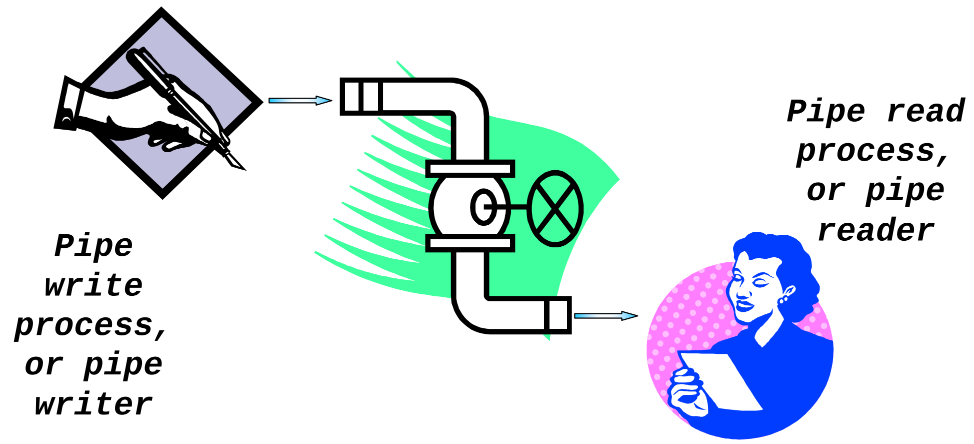
You can create UNIX named pipes from the shell or from an application using the mkfifo command or the mkfifo() function call. The named pipe remains in place until you delete it. You can use the named pipe as many times as you like. A named pipe is not bound to a specific process, so any number of processes can use the named pipe. You can also create a named pipe using the mknod command or mknod() function call.

You can list, rename, move, delete, and chmod named pipes using UNIX shell commands just as you can regular files.



Background information: UNIX named pipe characteristics

- Opened for writing and reading at the same time
- Data is stored in a buffer
- FIFO – First In, First Out

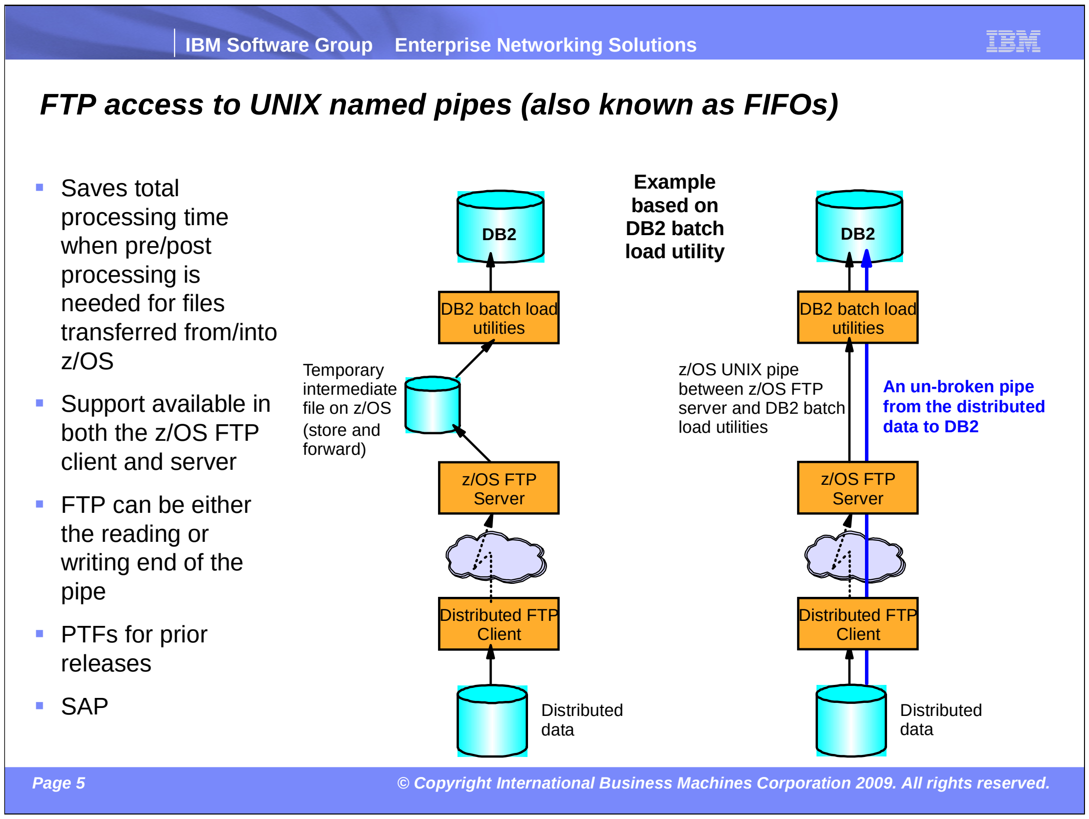

Pipe write process, or pipe writer
Pipe read process, or pipe reader

The process that will write to the pipe and the process that will read from the pipe must have the pipe open at the same time. You cannot open a pipe for writing, write to it, and close it, expecting another process to come along at its leisure to read the pipe sometime later. The first process to open a named pipe blocks the open() call until the other process opens the other end of the pipe.

As a UNIX pipe, any data that a process writes to a named pipe is not stored in the file system; the contents of a pipe reside in a finite buffer of volatile storage. A UNIX pipe fills up when the pipe writer writes more data than the pipe reader can read. Writing to a named pipe that is filled blocks the pipe writer until space frees up in the storage buffer.

A UNIX pipe is empty when the pipe reader reads and processes data faster than the pipe writer produces it. Reading from a named pipe that is empty blocks the pipe reader until the pipe writer writes into the named pipe.

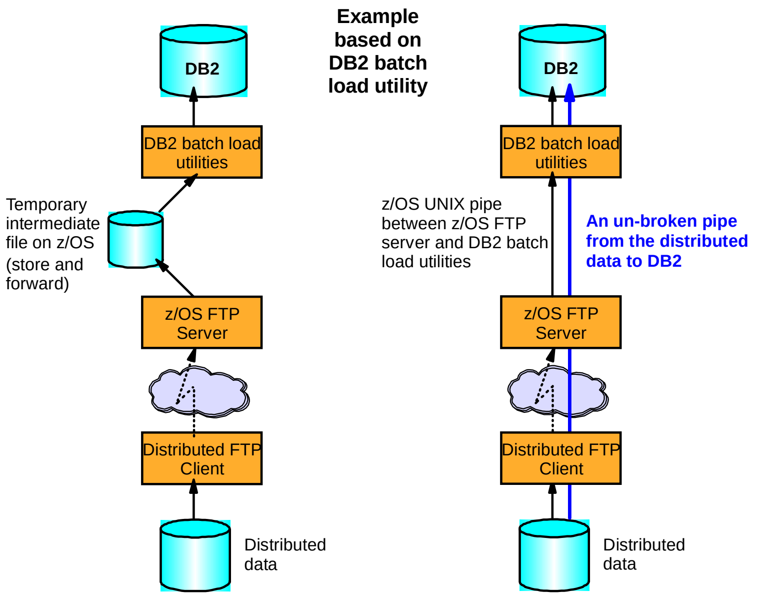
UNIX pipes are read and written strictly in FIFO (first in, first out) order. You cannot position the file marker to read in any other order. In fact, a common synonym for a named pipe is FIFO special file, or FIFO. Reading data from a named pipe removes the data from the pipe. The pipe reader cannot go back and read the same data from the pipe again. All writes to a named pipe are really appends to whatever is currently in the named pipe. It is impossible to replace the contents of a named pipe by writing to it.



IBM Software Group Enterprise Networking Solutions

FTP access to UNIX named pipes (also known as FIFOs)

- Saves total processing time when pre/post processing is needed for files transferred from/into z/OS
- Support available in both the z/OS FTP client and server
- FTP can be either the reading or writing end of the pipe
- PTFs for prior releases
- SAP


DB2
DB2 batch load utilities
Temporary intermediate file on z/OS (store and forward)
z/OS FTP Server
Distributed FTP Client
Distributed data
Example based on DB2 batch load utility
DB2
DB2 batch load utilities
z/OS UNIX pipe between z/OS FTP server and DB2 batch load utilities
An un-broken pipe from the distributed data to DB2
z/OS FTP Server
Distributed FTP Client
Distributed data

Page 5 © Copyright International Business Machines Corporation 2009. All rights reserved.

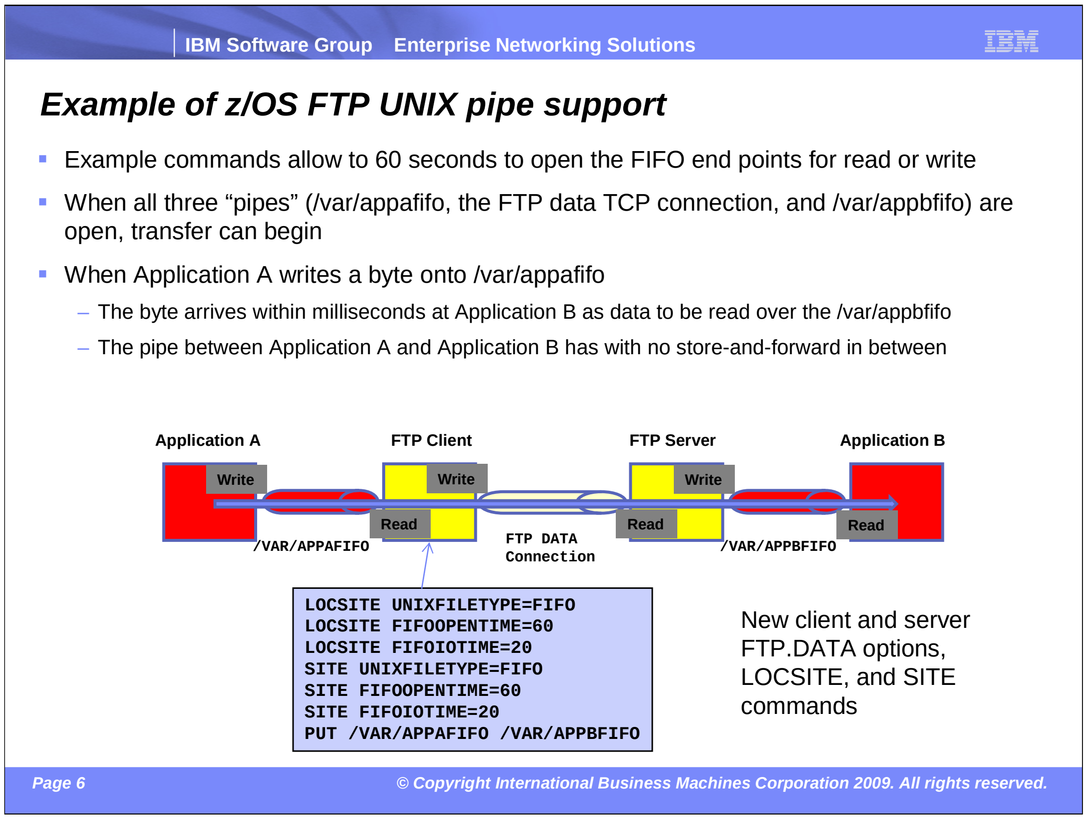
The z/OS FTP client and server are enhanced in z/OS V1R11 to write and read data directly into or from a z/OS UNIX named pipe.

The intention with this enhancement is better performance when files, for example, are transferred into z/OS and then need to be processed by a z/OS application. If that application supports reading from a z/OS UNIX named pipe, the transfer from the remote site can overlap processing by that z/OS application. That is, you have an unbroken pipe from the remote site that goes through a TCP connection and continues into a z/OS UNIX named pipe. When data is written from the remote client onto the TCP data connection, that data continues all the way into the post-processing application without further store and forward delays.

This z/OS FTP server support has been PTFed back to prior releases (z/OS V1R8, V1R9, and V1R10). The APAR number is PK71213.

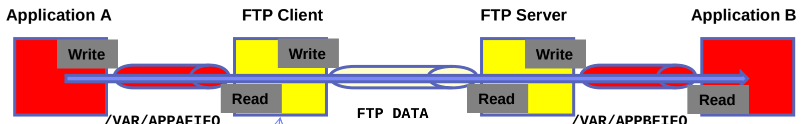
Combined with the z/OS FTP Client API, some very interesting integrated solutions can be developed. One example is splitting large files into multiple streams and transmitting them over parallel FTP sessions for improved performance. Another example is collecting information from multiple sources on z/OS, building a stream of data, and sending it directly to a remote site without ever storing it on z/OS.

The SAP on DB2® for z/OS Unicode FASTLOAD conversion utility exploits named pipes. This product migrates SAP installations to UNICODE. You can learn more about this SAP product and how it exploits named pipes by consulting the document *SAP Unicode Conversion on DB2 for z/OS – Cookbook.* Joachim Rese, IBM Germany, 04 November 2008.



Example of z/OS FTP UNIX pipe support

- Example commands allow to 60 seconds to open the FIFO end points for read or write
- When all three “pipes” (/var/appafifo, the FTP data TCP connection, and /var/appbfifo) are open, transfer can begin
- When Application A writes a byte onto /var/appafifo
  - The byte arrives within milliseconds at Application B as data to be read over the /var/appbfifo
  - The pipe between Application A and Application B has with no store-and-forward in between



```
LOCSITE UNIXFILETYPE=FIFO
LOCSITE FIFOOPENTIME=60
LOCSITE FIFOIOTIME=20
SITE UNIXFILETYPE=FIFO
SITE FIFOOPENTIME=60
SITE FIFOIOTIME=20
PUT /VAR/APPAFIFO /VAR/APPBFIFO
```

New client and server FTP.DATA options, LOCSITE, and SITE commands

This example is not necessarily realistic, but it is chosen to explain all the elements of the FTP pipe support.

The UNIXFILETYPE option applies only to files in the z/OS UNIX file system. UNIXFILETYPE tells FTP if such a name is a UNIX file (UNIXFILETYPE FILE which is the default) or a UNIX pipe (UNIXFILETYPE FIFO).

When you code the UNIXFILETYPE FILE statement in FTP.DATA, you are configuring FTP for the prior release behavior. FTP can create, store into, or send data from z/OS UNIX regular files, just as it has always done. When UNIXFILETYPE is FILE, you cannot use FTP to store into or send data from named pipes.

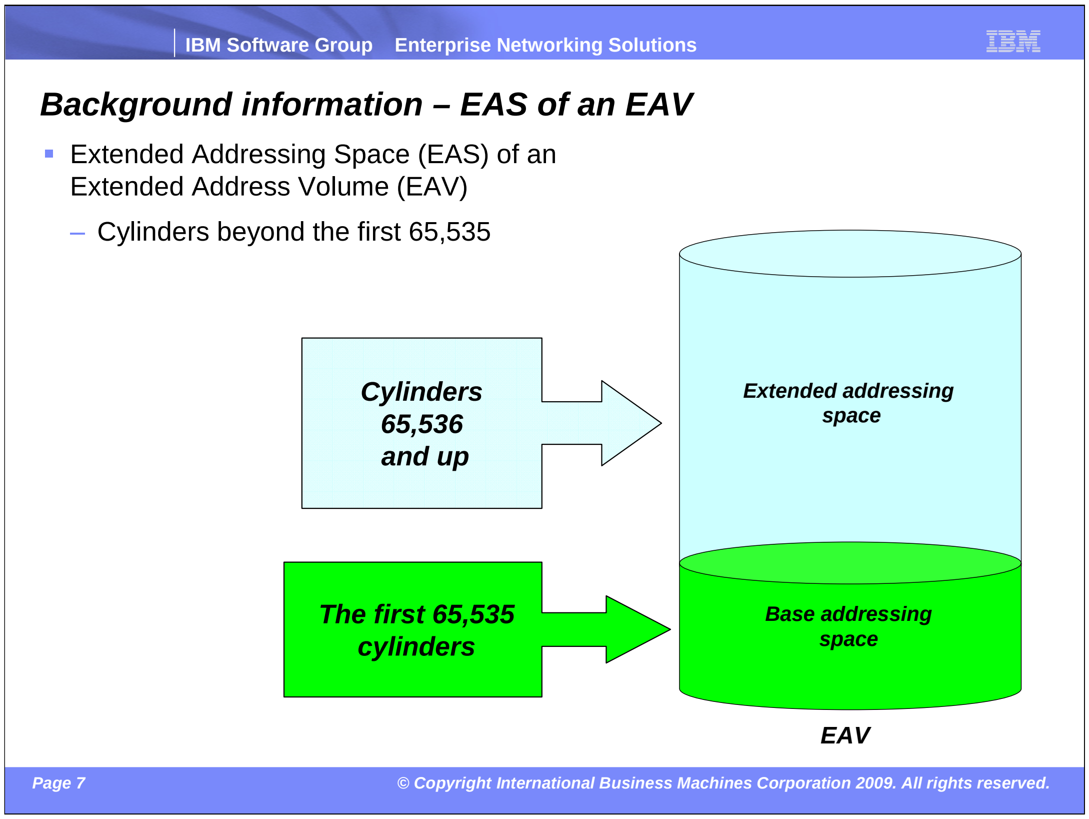
Code UNIXFILETYPE FIFO to configure FTP to access named pipes. With the FIFO option, FTP can store into and send data from z/OS UNIX named pipes, but not regular files.

You can change this value for the current session with the site and locsite subcommands. If your client or server is dedicated to named pipes, code UNIXFILETYPE FIFO in FTP.DATA to set the value for all sessions. Otherwise, a practical guideline is to code UNIXFILETYPE FILE in FTP.DATA (or let it take the default value), and to set it with a SITE or LOCSITE subcommand to FIFO just before accessing named pipes.

You cannot open a named pipe for writing until another process opens the named pipe for reading; you cannot open a named pipe for reading until another process opens the named pipe for writing. The first process to open the named pipe will block until the other process opens the named pipe.

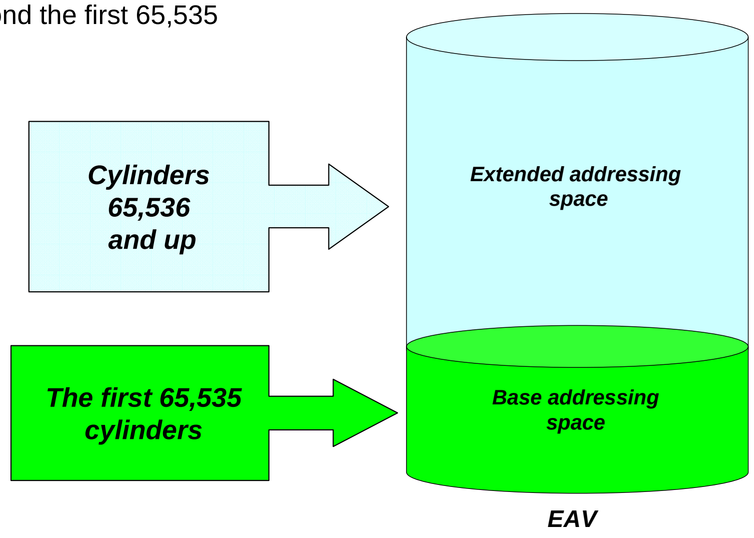
You can control the amount of time z/OS FTP waits for the other process to open a named pipe by configuring the FIFOOPENTIME value. When FTP opens a named pipe, it will wait up to FIFOOPENTIME for another process to open the named pipe. If no other process opens the named pipe in FIFOOPENTIME seconds, FTP will fail the file transfer.

When FTP reads from a named pipe, it will wait up to FIFOIOTIME seconds for the read to complete. If the read does not complete within FIFOIOTIME seconds (because the other process has not written any data for that amount of time), FTP will fail the transfer. When FTP stores into a named pipe, it will wait up to FIFOIOTIME seconds for the write to complete. If the write does not complete in FIFOIOTIME seconds, FTP will fail the transfer.



Background information – EAS of an EAV

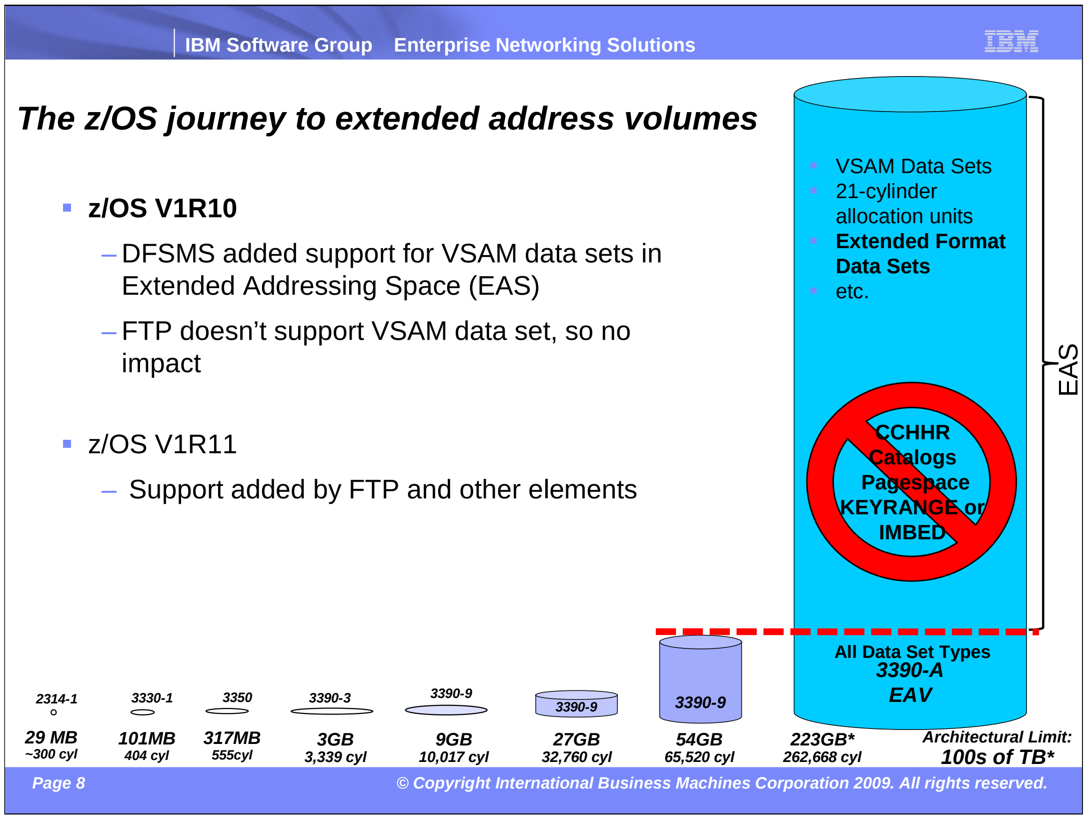
- Extended Addressing Space (EAS) of an Extended Address Volume (EAV)
  - Cylinders beyond the first 65,535



The capacity of storage volumes has increased steadily over the years within the constraints of the 3390 storage device architecture. As of z/OS V1R10, z/OS has added support for DASD volumes having more than 65,520 cylinders. An Extended Address Volume (EAV) is a DASD volume that exceeds 65,520 cylinders. The DASD volume maximum for V1R10 is 262,668 cylinders.

The EAS of an EAV is an abbreviation for Extended Addressing Space (EAS) of an EAV. It refers to those cylinders of a volume beyond the first 65,535.

In this diagram of an EAV, cylinders up to but not including cylinder 65,536 are in the base addressing space of the EAV. Cylinders starting with cylinder 65,536 are in the EAS of the EAV.


IBM Software Group Enterprise Networking Solutions

The z/OS journey to extended address volumes

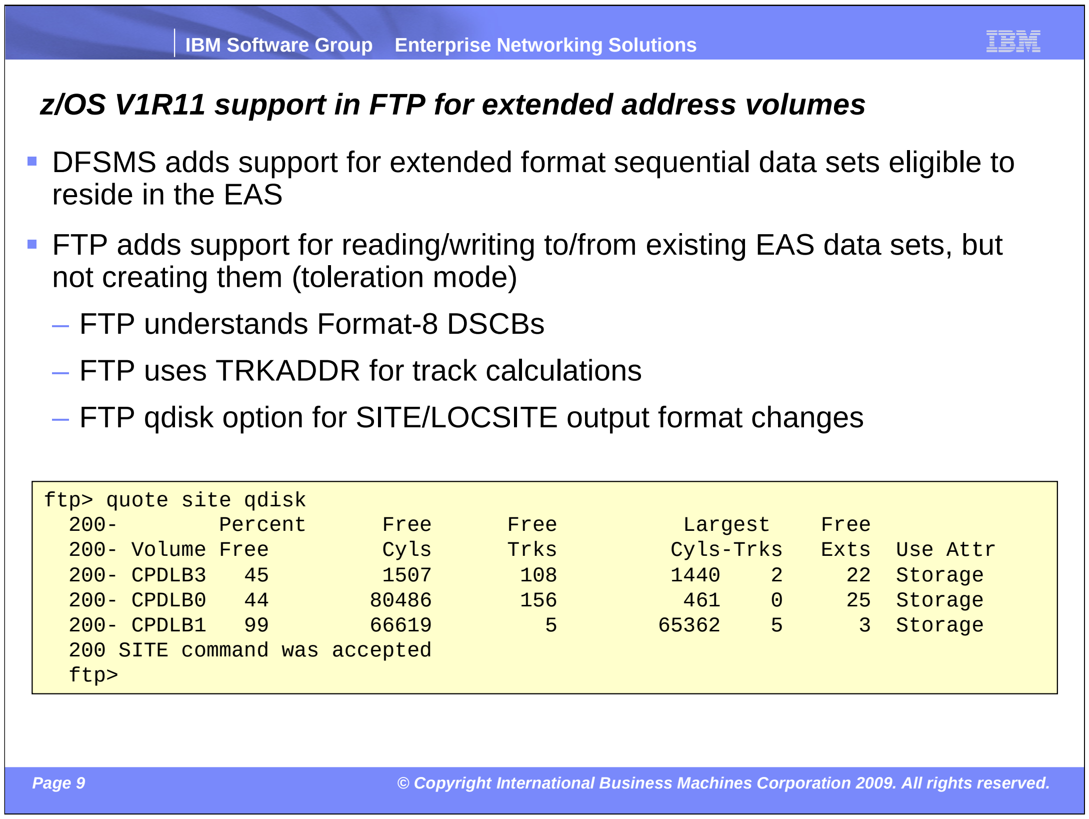
- **z/OS V1R10**
  - DFSMS added support for VSAM data sets in Extended Addressing Space (EAS)
  - FTP doesn't support VSAM data set, so no impact
- z/OS V1R11
  - Support added by FTP and other elements

VSAM Data Sets
21-cylinder allocation units
Extended Format Data Sets
etc.
EAS
CCHHR
Catalogs
Pagespace
KEYRANGE or IMBED
All Data Set Types
3390-A EAV
2314-1
3330-1
3350
3390-3
3390-9
3390-9
3390-9
29 MB ~300 cyl
101MB 404 cyl
317MB 555cyl
3GB 3,339 cyl
9GB 10,017 cyl
27GB 32,760 cyl
54GB 65,520 cyl
223GB* 262,668 cyl
Architectural Limit: 100s of TB*

Page 8 © Copyright International Business Machines Corporation 2009. All rights reserved.

DFSMS is adding support over several releases for data sets in the extended addressing space (EAS) of extended address volumes (EAV). Initially, DFSMS only supported VSAM data sets in the EAS. In z/OS V1R11, DFSMS also supports extended format sequential data sets (not normal sequential data sets) in the EAS.

z/OS V1R11 adds some support for EAV extensions, including Extended Format sequential data sets and data sets spanning the 64K cylinder line. There is a new JCL keyword and data set attribute (EATTR). SMS, SMF, TSO/E, ISPF, and FTP support EAV extensions. Language Environment supports data sets with extended attributes, including those in the EAS on an EAV, for C/C++ programs. RACF® has discrete profile support for non-VSAM data sets in EAS. And SVCDUMP, SYSMDUMP, SNAP, ABDUMP, and IPCS support EAV extensions.



z/OS V1R11 support in FTP for extended address volumes

- DFSMS adds support for extended format sequential data sets eligible to reside in the EAS
- FTP adds support for reading/writing to/from existing EAS data sets, but not creating them (toleration mode)
  - FTP understands Format-8 DSCBs
  - FTP uses TRKADDR for track calculations
  - FTP qdisk option for SITE/LOCSITE output format changes

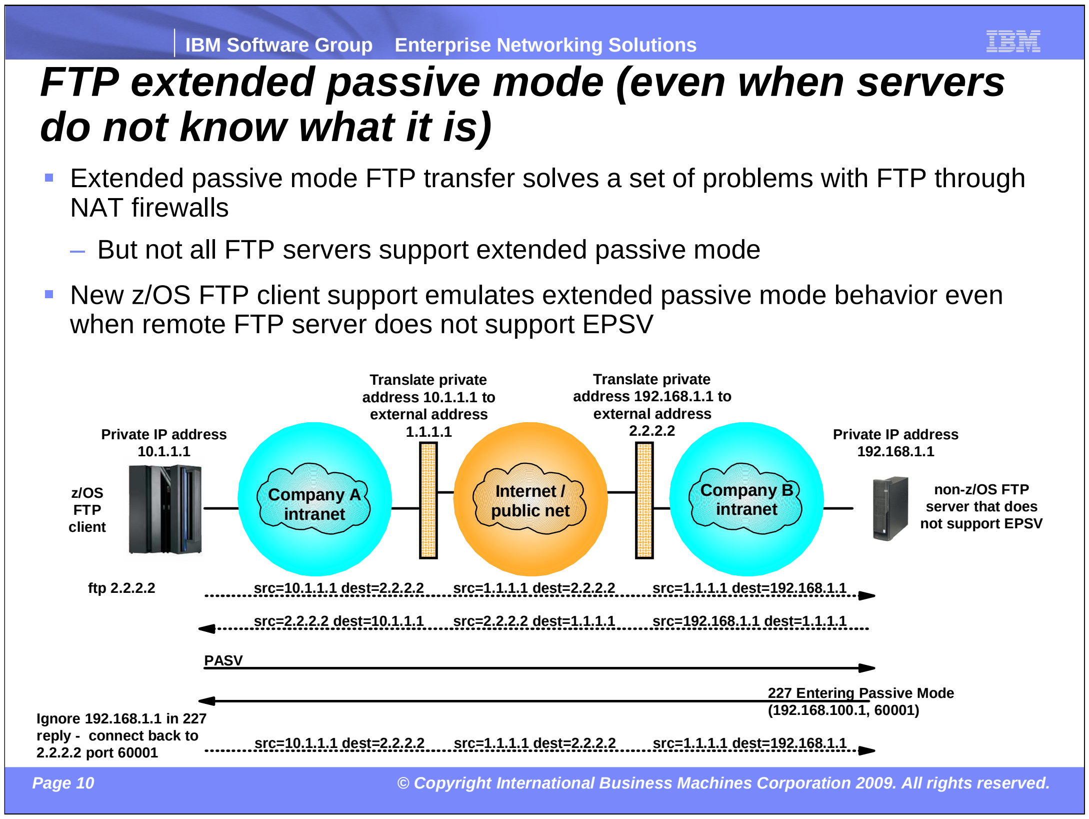
```
ftp> quote site qdisk
  200-        Percent       Free      Free          Largest     Free
  200- Volume Free          Cyls      Trks         Cyls-Trks    Exts  Use Attr
  200- CPDLB3   45          1507       108          1440    2     22  Storage
  200- CPDLB0   44         80486       156           461    0     25  Storage
  200- CPDLB1   99         66619         5         65362    5      3  Storage
  200 SITE command was accepted
  ftp>
```

In z/OS V1R11, DFSMS adds support for extended format sequential data sets (not normal sequential data sets). Data sets that are eligible to reside in the EAS are allocated by DFSMS using a format-8 DSCB.

FTP reads and writes to such data sets in the EAS. FTP does not provide explicit configuration options to allocate data sets in the EAS or to specify data sets as eligible to reside in the EAS (unless implicitly done by way of SMS data class assignments).

FTP provides information about EAV volumes on the qdisk command. If you screen-scrape qdisk output, note that you will have to modify the screen-scraping logic since the qdisk output was changed between V1R10 and V1R11 to accommodate EAVs.

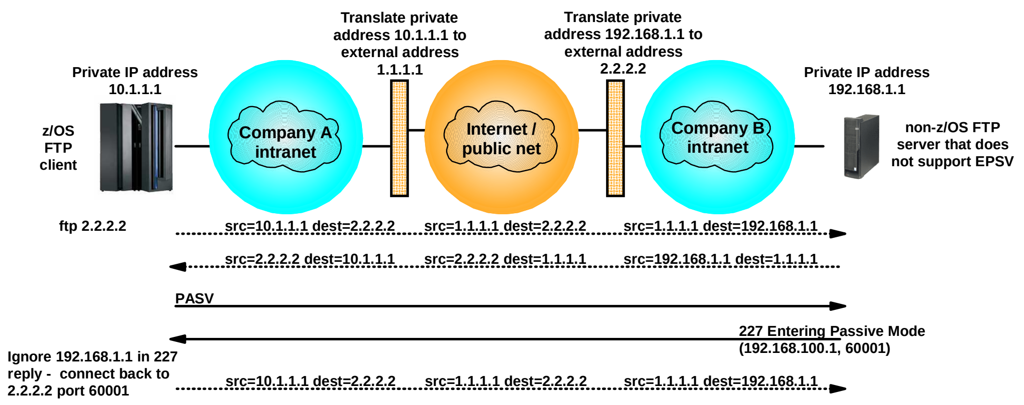
The LIST command output format did NOT change. However, if an output value is larger than the space allowed on the line, plus signs are displayed instead of the actual value.



IBM Software Group | Enterprise Networking Solutions

FTP extended passive mode (even when servers do not know what it is)

- Extended passive mode FTP transfer solves a set of problems with FTP through NAT firewalls
  - But not all FTP servers support extended passive mode
- New z/OS FTP client support emulates extended passive mode behavior even when remote FTP server does not support EPSV


Translate private address 10.1.1.1 to external address 1.1.1.1
Translate private address 192.168.1.1 to external address 2.2.2.2
Private IP address 10.1.1.1
Private IP address 192.168.1.1
z/OS FTP client
Company A intranet
Internet / public net
Company B intranet
non-z/OS FTP server that does not support EPSV
ftp 2.2.2.2
src=10.1.1.1 dest=2.2.2.2
src=1.1.1.1 dest=2.2.2.2
src=1.1.1.1 dest=192.168.1.1
src=2.2.2.2 dest=10.1.1.1
src=2.2.2.2 dest=1.1.1.1
src=192.168.1.1 dest=1.1.1.1
PASV
227 Entering Passive Mode (192.168.100.1, 60001)
Ignore 192.168.1.1 in 227 reply - connect back to 2.2.2.2 port 60001
src=10.1.1.1 dest=2.2.2.2
src=1.1.1.1 dest=2.2.2.2
src=1.1.1.1 dest=192.168.1.1

Page 10 © Copyright International Business Machines Corporation 2009. All rights reserved.

In z/OS V1R11, the z/OS FTP client implements an option that allows the FTP client to simulate the use of extended passive mode - even in cases where the remote FTP server does not support EPSV. The option enables passive mode data connections to succeed when the server is behind Network Address Translation (NAT) processing and might not advertise its correct external IP address to the client.

This option instructs the z/OS FTP client to ignore the IP address it receives in the PASV reply from the server. Also with this option, the z/OS FTP client uses the same IP address for the data connection that was used for the control connection.

The option is PASSIVEIGNOREADDR in the FTP.DATA configuration file. When PASSIVEIGNOREADDR is enabled, the client will ignore the IP address presented in the reply and use the same address it used to log into the FTP server. For the port, it will still use the port received in the PASV reply. PASSIVEIGNOREADDR can be either TRUE or FALSE. By default it is FALSE to prevent migration problems.

Also, FTP supports PASSIVEIGNOREADDR and NOPASSIVEIGNOREADDR options on the LOCsite command. The default is NOPASSIVEIGNOREADDR.

The FWFRIENDLY parameter must be set to TRUE for the PASSIVEIGNOREADDR option to have an effect.

These enhancements generally allow passive mode FTP data connections to be established through NAT firewalls in cases in which EPSV (extended passive mode support) established connections. In some cases, these enhancements might not be sufficient to allow the connections to be established through the NAT firewalls. These cases are when the FTP control connection is secured with SSL/TLS and the NAT firewalls (in addition to performing NAT) also implement dynamic filters based on the content of the PASV (or EPSV) reply. In these cases, use the FTP CCC command (clear command channel).

Trademarks, copyrights, and disclaimers