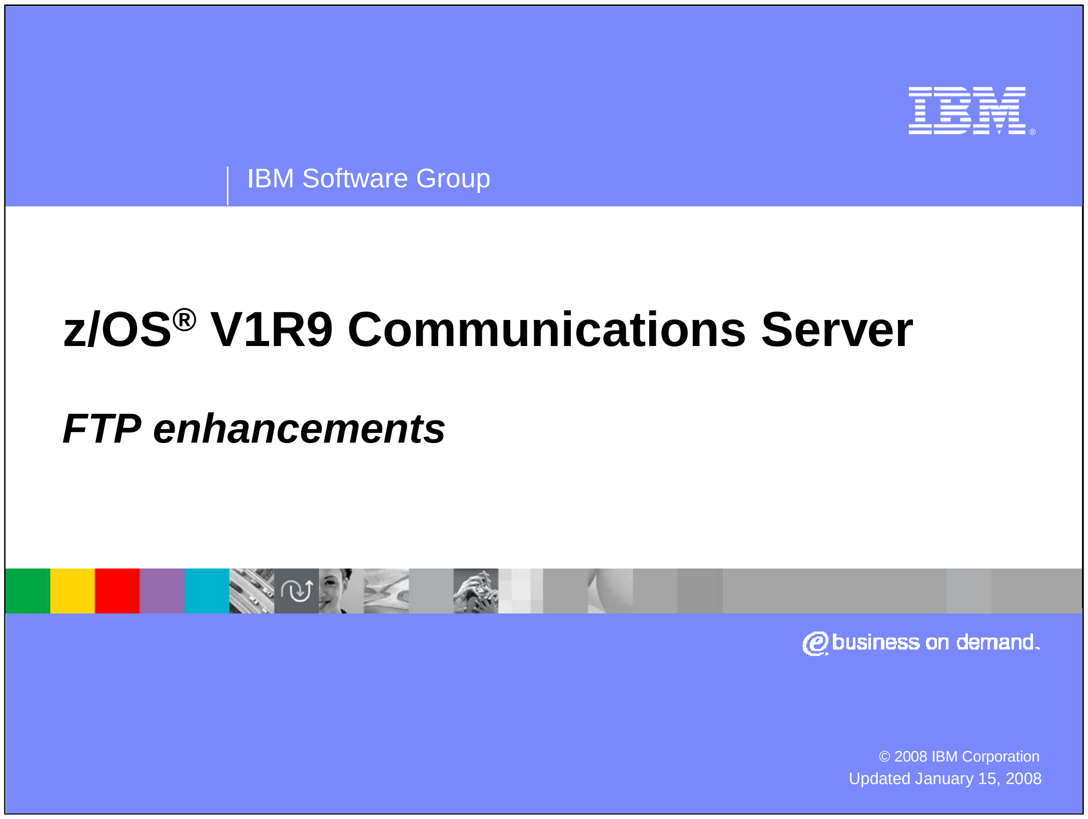


IBM Software Group

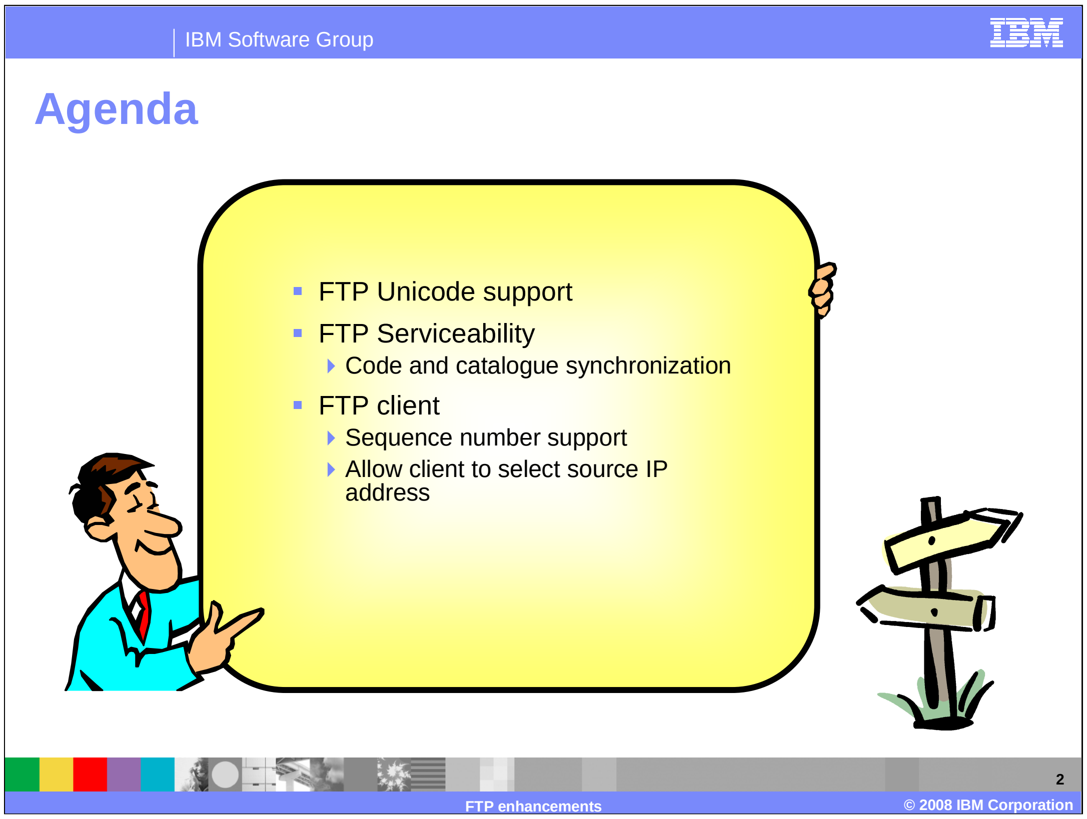
z/OS® V1R9 Communications Server

FTP enhancements

business on demand.

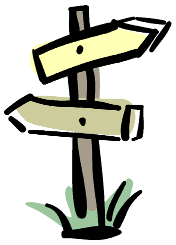
Updated January 15, 2008

This presentation is an overview of FTP Enhancements for the z/OS V1R9 Communications Server.

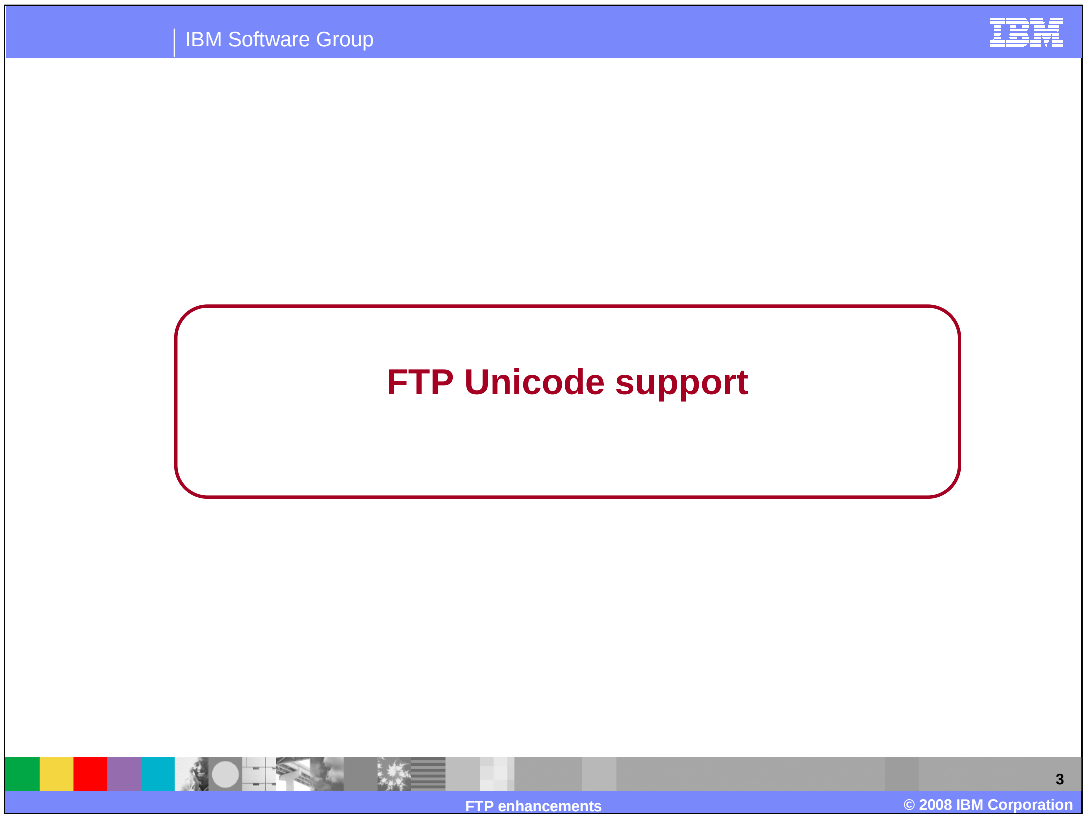


Agenda

- FTP Unicode support
- FTP Serviceability
  - Code and catalogue synchronization
- FTP client
  - Sequence number support
  - Allow client to select source IP address

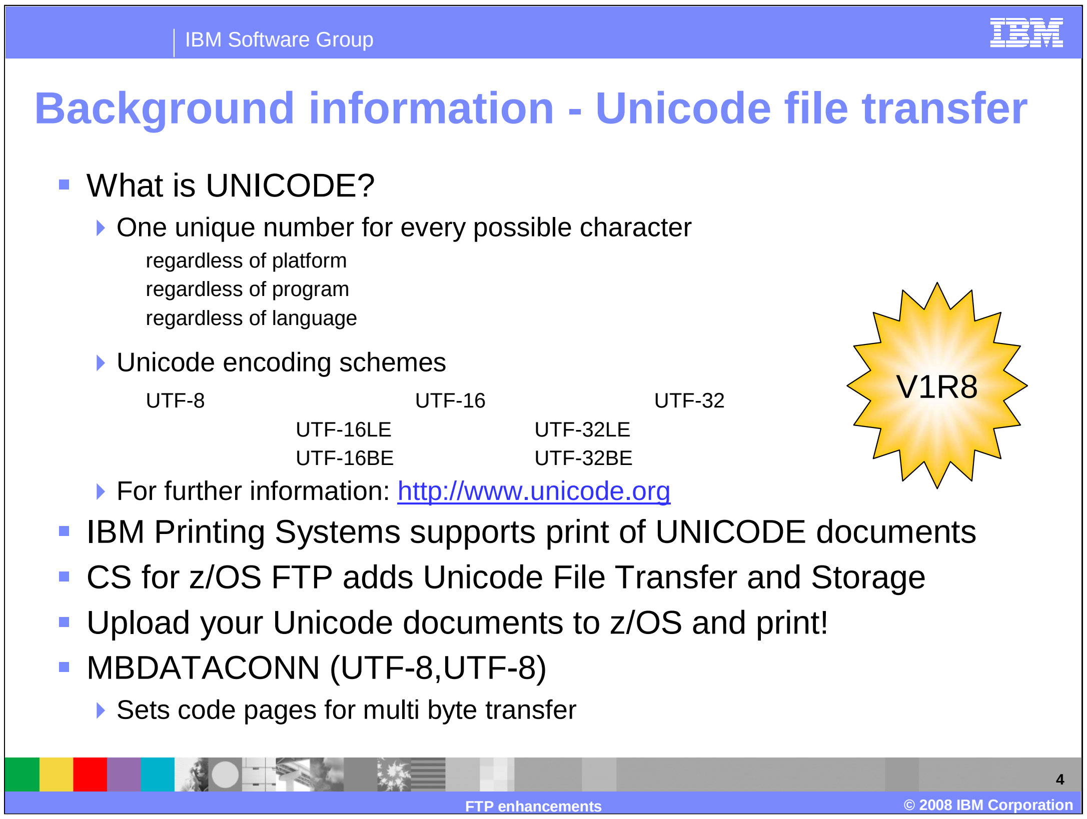


This presentation covers the Unicode support, a serviceability enhancement and enhancements that affect the FTP client only.



FTP Unicode support

This section describes the enhancements made in the z/OS V1R9 Communications Server to z/OS FTP Unicode file transfer and storage.


IBM Software Group

Background information - Unicode file transfer

- What is UNICODE?
  - One unique number for every possible character
    - regardless of platform
    - regardless of program
    - regardless of language
  - Unicode encoding schemes

UTF-8	UTF-16	UTF-32
	UTF-16LE	UTF-32LE
	UTF-16BE	UTF-32BE

  - For further information: http://www.unicode.org
- IBM Printing Systems supports print of UNICODE documents
- CS for z/OS FTP adds Unicode File Transfer and Storage
- Upload your Unicode documents to z/OS and print!
- MBDATACONN (UTF-8,UTF-8)
  - Sets code pages for multi byte transfer

V1R8

4

FTP enhancements

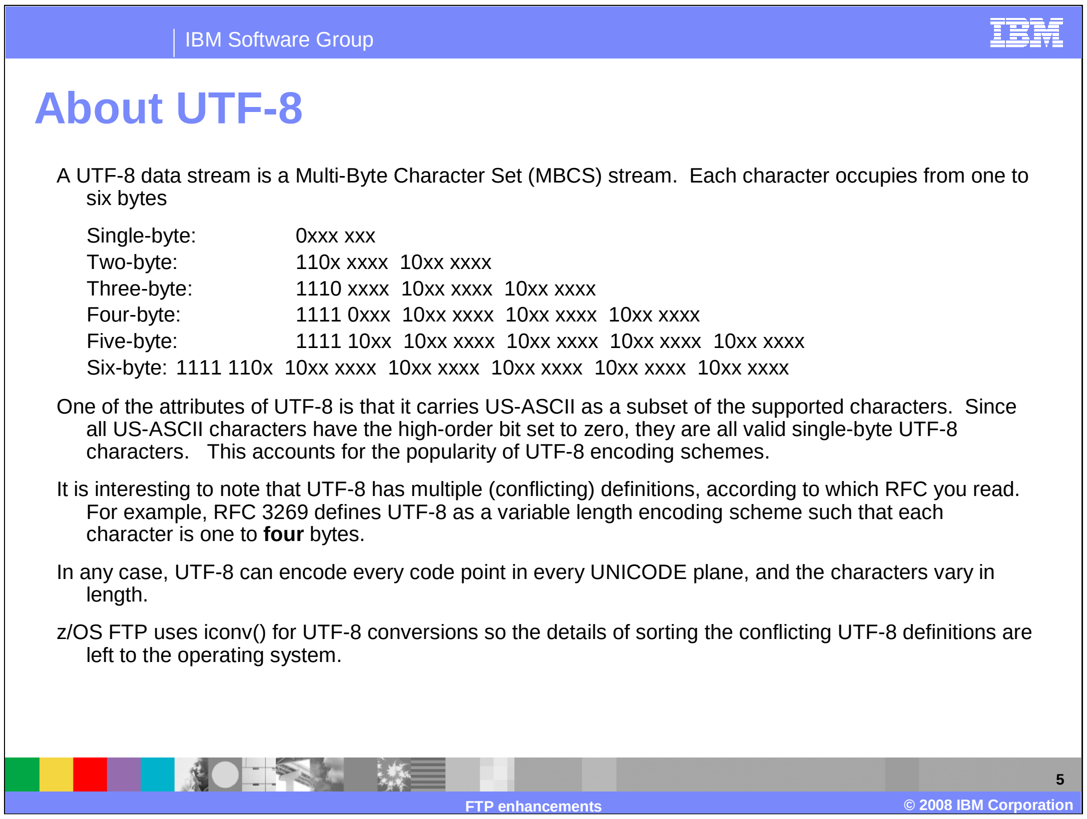
© 2008 IBM Corporation

Unicode is defined by the Unicode Consortium. Their goal is to is to define encoding schemes that have the ability to encode every possible character in the universe. The URL in this slide is the Web page of the Unicode Consortium. It is an excellent resource for learning about Unicode.

In z/OS V1R8 Communications Server, to support IBM Printing System's new support for UNICODE documents, CS for z/OS added Unicode File Transfer and storage. You can now move UNICODE documents to a z/OS host to store and to print.

In z/OS V1R8 Communications Server, z/OS FTP enhanced the configuration option, MBDATACONN, to support Unicode file transfer and storage. The MBDATACONN configuration option is used to specify which code pages to use for multi byte transfer. The first code page is the file storage code page; the second is the network transfer code page. These code pages are supported in pairs; the *IP Configuration Reference*, MBDATACONN statement, lists the code page pairs that are supported for multi byte transfer. The pair (UTF-8,UTF-8) was added in V1R8. This means that you can transfer and store a UTF-8 file. UTF-8 is one of the Unicode encodings.

Details about the different Unicode encoding schemes follow.



About UTF-8

A UTF-8 data stream is a Multi-Byte Character Set (MBCS) stream. Each character occupies from one to six bytes

Single-byte:	0xxx xxx
Two-byte:	110x xxxx 10xx xxxx
Three-byte:	1110 xxxx 10xx xxxx 10xx xxxx
Four-byte:	1111 0xxx 10xx xxxx 10xx xxxx 10xx xxxx
Five-byte:	1111 10xx 10xx xxxx 10xx xxxx 10xx xxxx 10xx xxxx
Six-byte:	1111 110x 10xx xxxx 10xx xxxx 10xx xxxx 10xx xxxx 10xx xxxx

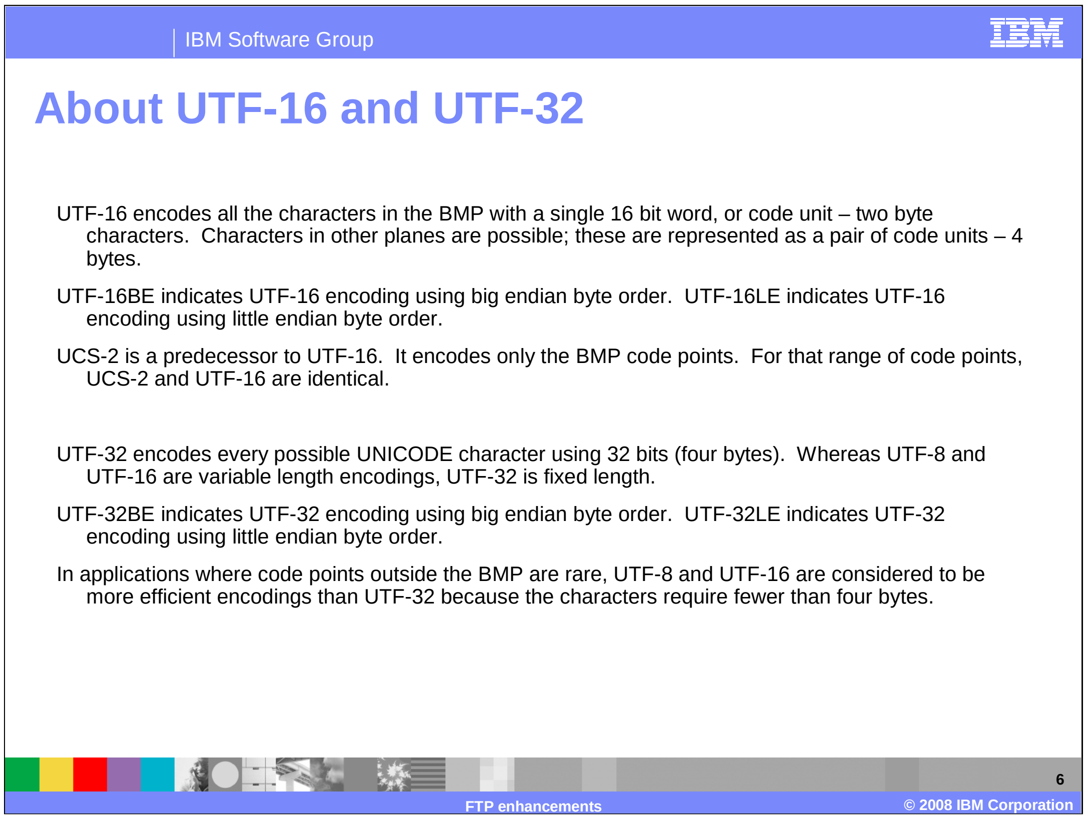
One of the attributes of UTF-8 is that it carries US-ASCII as a subset of the supported characters. Since all US-ASCII characters have the high-order bit set to zero, they are all valid single-byte UTF-8 characters. This accounts for the popularity of UTF-8 encoding schemes.

It is interesting to note that UTF-8 has multiple (conflicting) definitions, according to which RFC you read. For example, RFC 3269 defines UTF-8 as a variable length encoding scheme such that each character is one to **four** bytes.

In any case, UTF-8 can encode every code point in every UNICODE plane, and the characters vary in length.

z/OS FTP uses iconv() for UTF-8 conversions so the details of sorting the conflicting UTF-8 definitions are left to the operating system.

These notes provide more information about the UTF-8 encoding.



About UTF-16 and UTF-32

UTF-16 encodes all the characters in the BMP with a single 16 bit word, or code unit – two byte characters. Characters in other planes are possible; these are represented as a pair of code units – 4 bytes.

UTF-16BE indicates UTF-16 encoding using big endian byte order. UTF-16LE indicates UTF-16 encoding using little endian byte order.

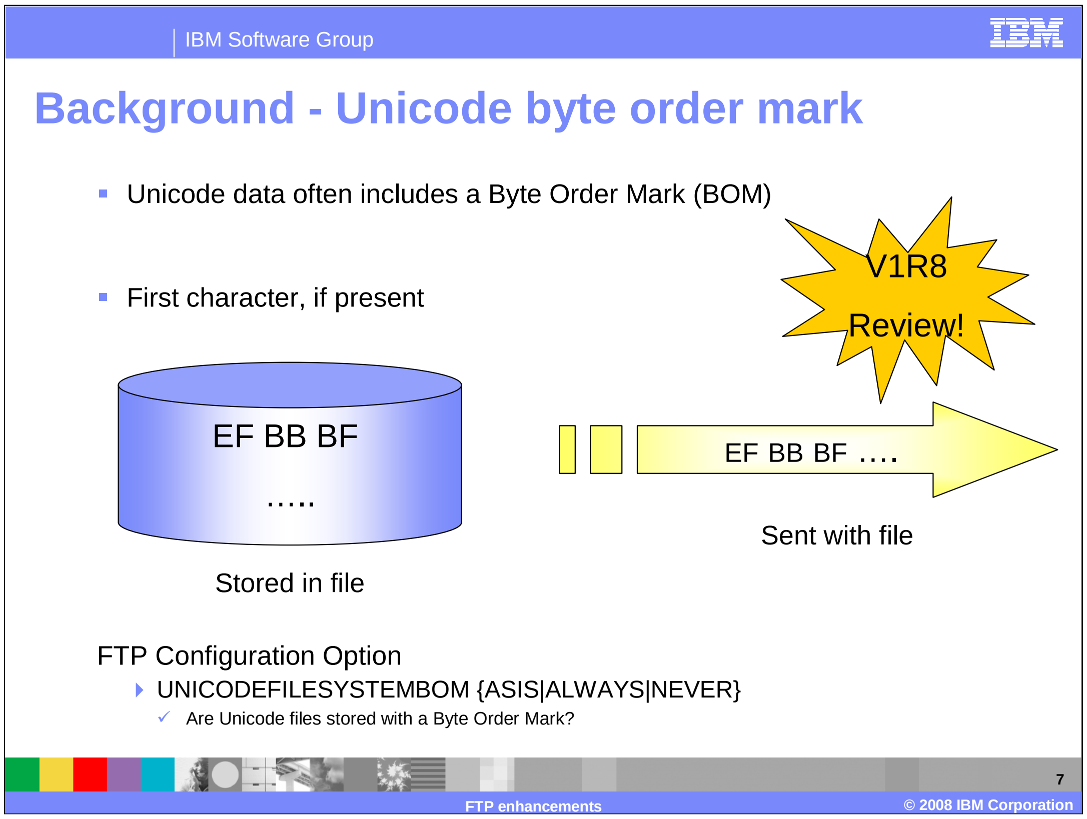
UCS-2 is a predecessor to UTF-16. It encodes only the BMP code points. For that range of code points, UCS-2 and UTF-16 are identical.

UTF-32 encodes every possible UNICODE character using 32 bits (four bytes). Whereas UTF-8 and UTF-16 are variable length encodings, UTF-32 is fixed length.

UTF-32BE indicates UTF-32 encoding using big endian byte order. UTF-32LE indicates UTF-32 encoding using little endian byte order.

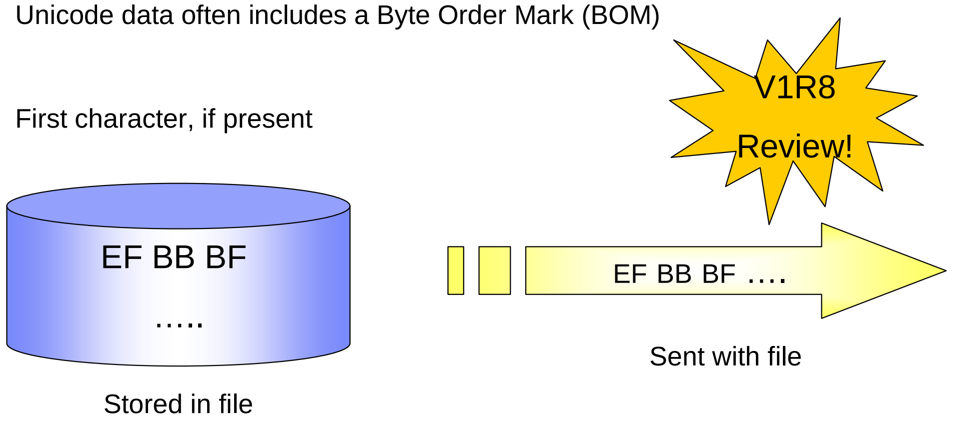
In applications where code points outside the BMP are rare, UTF-8 and UTF-16 are considered to be more efficient encodings than UTF-32 because the characters require fewer than four bytes.

These notes provide some information about UTF-16 and UTF-32.



Background - Unicode byte order mark

- Unicode data often includes a Byte Order Mark (BOM)
- First character, if present

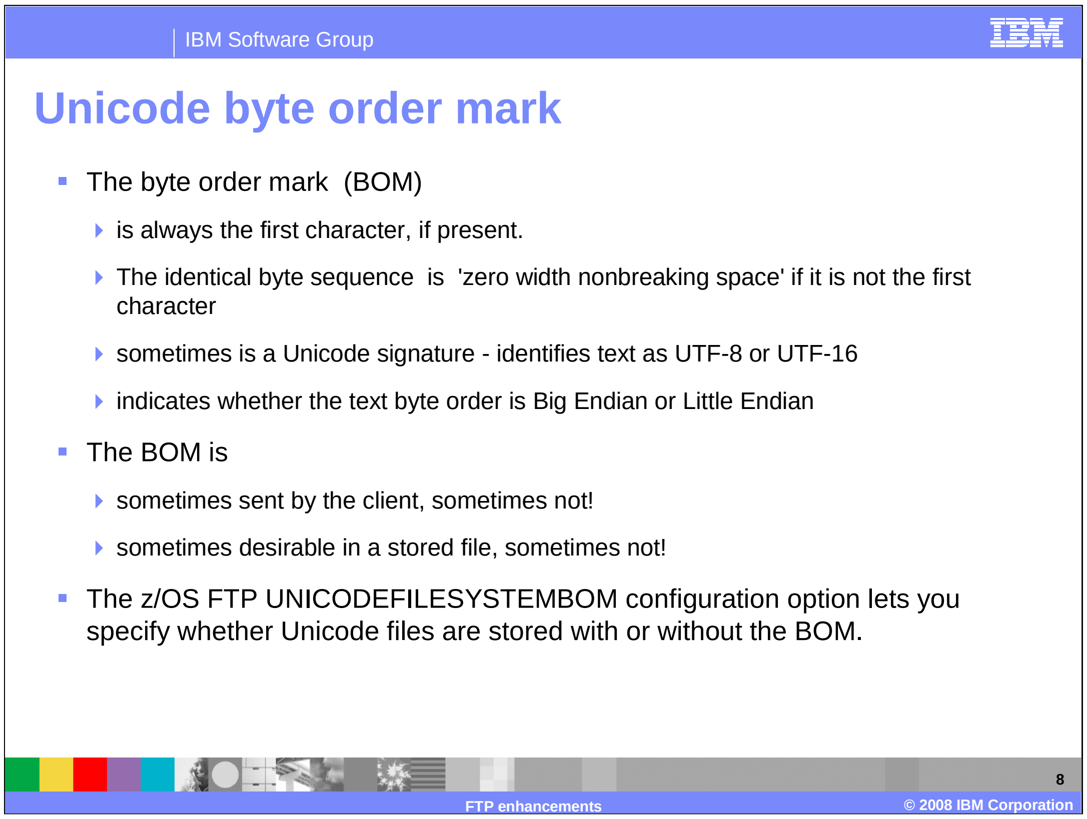


FTP Configuration Option
- UNICODEFILESYSTEMBOM {ASIS|ALWAYS|NEVER}
  - ✓ Are Unicode files stored with a Byte Order Mark?

Unicode files and Unicode data streams often include a Byte Order Mark (BOM). The BOM can be stored in a Unicode file, and it can be sent as part of the file when uploading or downloading the file with FTP.

If it is present in the file or data stream, it will be the first character. The byte sequence you see in this slide is the UTF-8 BOM. Each Unicode encoding has a different BOM. Your notes have a chart of the BOM for each Unicode encoding. Right now, the important thing to remember is that the BOM might appear in the data, or might not.

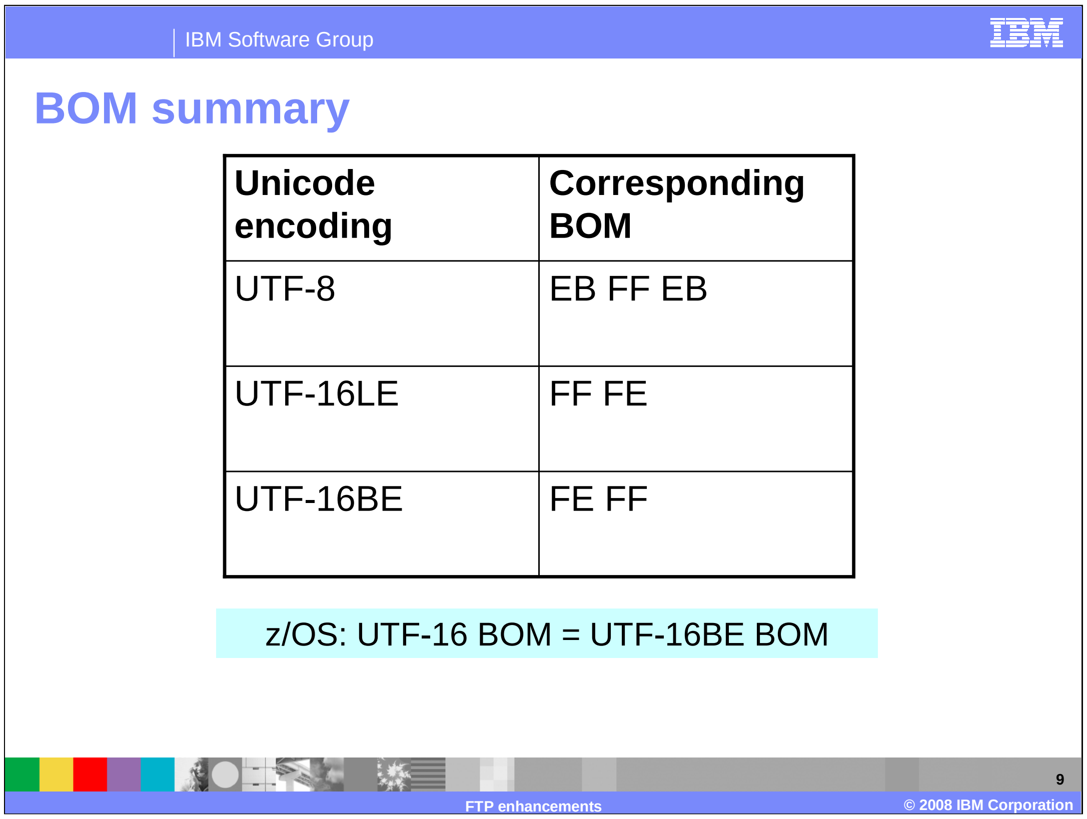
In the z/OS V1R8 Communications Server, z/OS FTP introduced a new configuration option, UNICODEFILESYSTEMBOM. The UNICODEFILESYSTEMBOM configuration option controls whether an inbound Unicode file is stored with a BOM. As you might expect, ALWAYS means always store the file with a BOM; NEVER means never store the file with a BOM. ASIS means to store the file with a BOM if it was sent with a BOM.



Unicode byte order mark

- The byte order mark (BOM)
  - is always the first character, if present.
  - The identical byte sequence is 'zero width nonbreaking space' if it is not the first character
  - sometimes is a Unicode signature - identifies text as UTF-8 or UTF-16
  - indicates whether the text byte order is Big Endian or Little Endian
- The BOM is
  - sometimes sent by the client, sometimes not!
  - sometimes desirable in a stored file, sometimes not!
- The z/OS FTP UNICODEFILESYSTEMBOM configuration option lets you specify whether Unicode files are stored with or without the BOM.

The z/OS V1R8 Communications Server added the UNICODEFILESYSTEMBOM statement to configure whether to store the BOM when storing an incoming Unicode file.



IBM Software Group

BOM summary

Unicode encoding	**Corresponding BOM**
UTF-8	EB FF EB
UTF-16LE	FF FE
UTF-16BE	FE FF

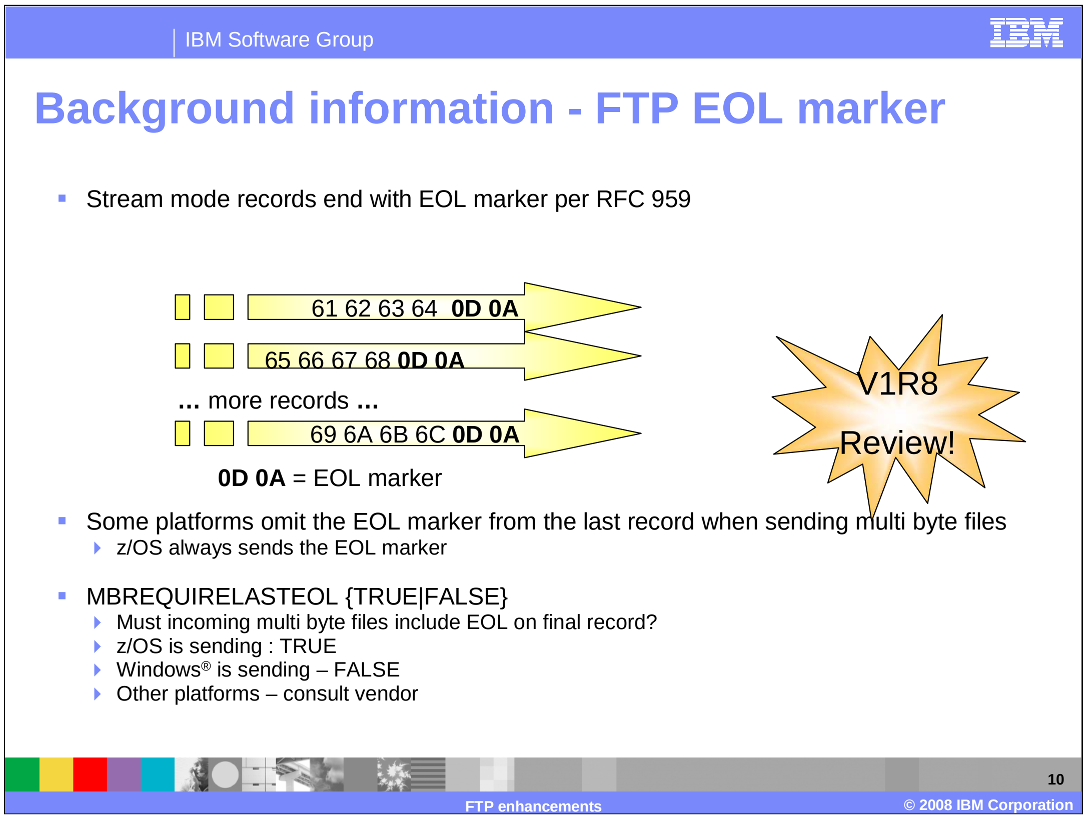
z/OS: UTF-16 BOM = UTF-16BE BOM

9

FTP enhancements

© 2008 IBM Corporation

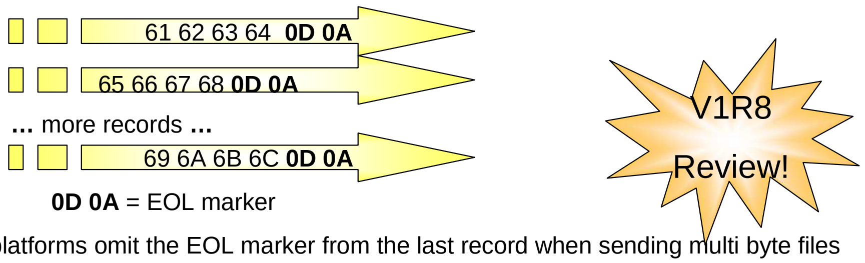
For each of the Unicode code pages, the corresponding Byte Order Mark is given.



IBM Software Group

Background information - FTP EOL marker

- Stream mode records end with EOL marker per RFC 959



- Some platforms omit the EOL marker from the last record when sending multi byte files
  - z/OS always sends the EOL marker
- MBREQUIRELASTEOL {TRUE|FALSE}
  - Must incoming multi byte files include EOL on final record?
  - z/OS is sending : TRUE
  - Windows® is sending – FALSE
  - Other platforms – consult vendor

10

FTP enhancements

© 2008 IBM Corporation

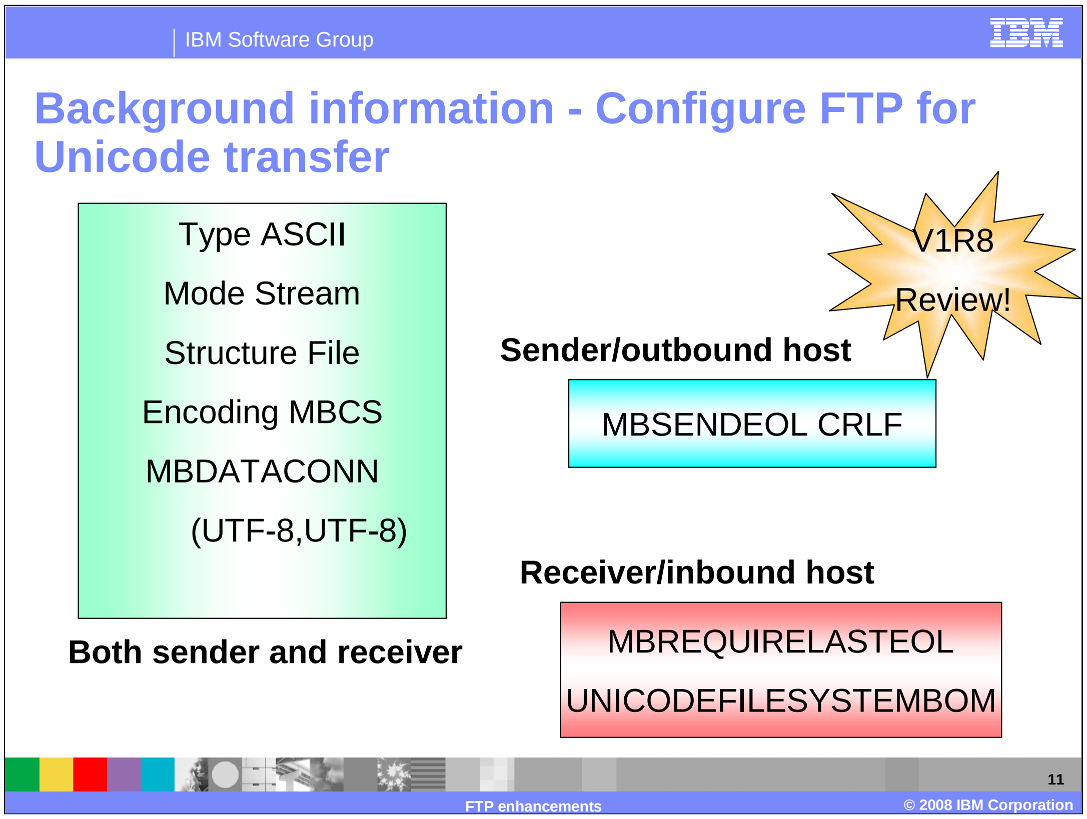
CS for z/OS FTP added the MBREQUIRELASTEOL configuration option. This applies to all multi byte transfers, not just to Unicode transfers.

To understand MBREQUIRELASTEOL, recall that whenever FTP sends a file as type ASCII, structure File, in stream mode, each record ends in an End of Line (EOL) marker. This is true regardless of whether you are sending single byte or multi byte data.

In the diagram above, FTP is sending multiple lines in stream mode. The encoding is UTF-8. It looks like ASCII because seven bit ASCII is a subset of UTF-8, and all the characters of the example are in the ASCII range. The text is letters of the alphabet. The important thing to notice here is that each record sent ends with the EOL marker, x’0D 0A’. This is as specified in RFC 959.

When sending a multi byte file, some FTP implementations include an EOL marker on the final record, and some do not. z/OS always includes the last EOL. Windows does not add an EOL to the last record when sending multi byte files.

For incoming multi byte files, use the MBREQUIRELASTEOL configuration option at the receiver to specify what z/OS should expect from the sender.


IBM Software Group

Background information - Configure FTP for Unicode transfer

Type ASCII

Mode Stream

Structure File

Encoding MBCS

MBDATACONN

(UTF-8,UTF-8)

Both sender and receiver

V1R8

Review!

Sender/outbound host

MBSENDEOL CRLF

Receiver/inbound host

MBREQUIRELASTEOL

UNICODEFILESYSTEMBOM

11

FTP enhancements

© 2008 IBM Corporation

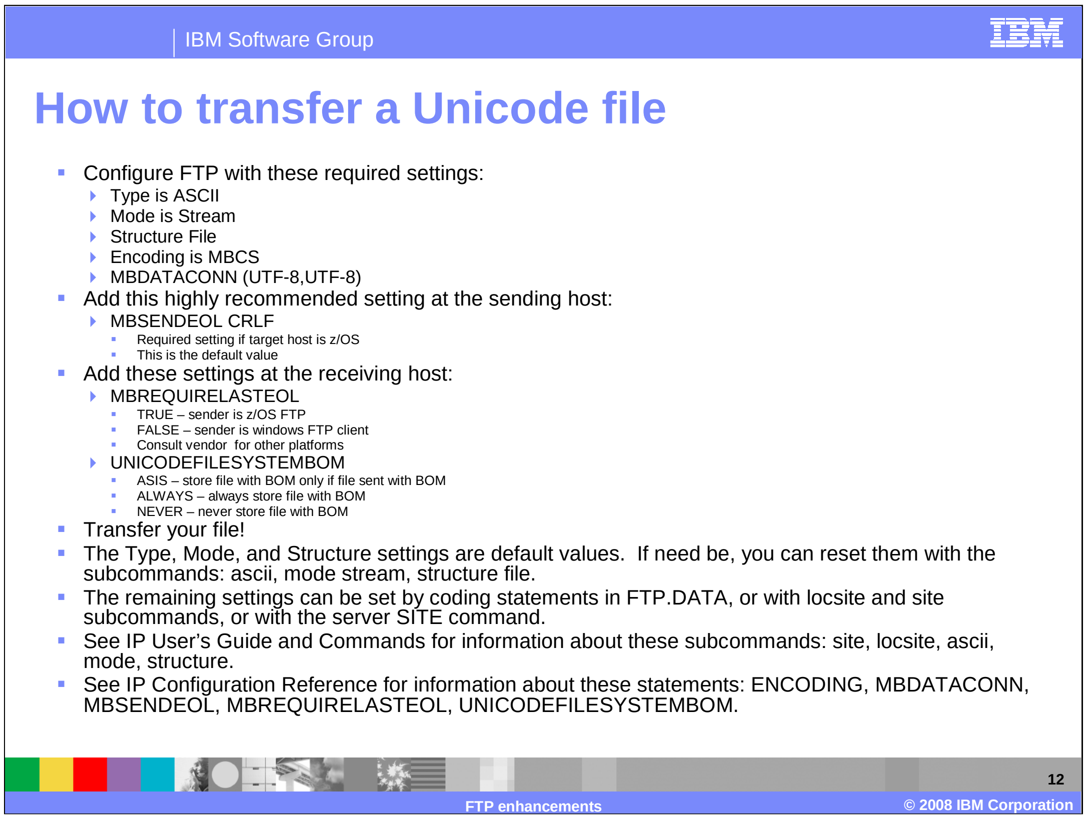
Here is a quick reminder of how to configure FTP to transfer a Unicode file using the z/OS V1R8 Communications Server FTP support for Unicode File Transfer and Storage.

The box on the left in green shows the configuration settings that must be set on both the sending and receiving hosts; the cyan box on the upper right shows the configuration settings that must be set at the sending host; the red box on the lower right shows the configuration options that must be set at the receiving host.

MBSENDEOL was introduced in V1R7, and applies only to outbound multi byte transfer. Although alternate settings for MBSENDEOL are available, most users should stick with CRLF, the default value and the value specified in the FTP standard, RFC 959.   This is the only setting that works when sending to a z/OS host.

MBREQUIRELASTEOL and UNICODEFILESYSTEMBOM were introduced in V1R8, and apply only to inbound multi byte file transfer.   You have to understand the sending host's FTP implementation to set MBREQUIRELASTEOL correctly; the file transfer will fail if you pick the wrong value.   That's the bad news; the good news is that you can toggle to the other setting and try again if you get it wrong, or you can consult the notes on the next page for suggested values.

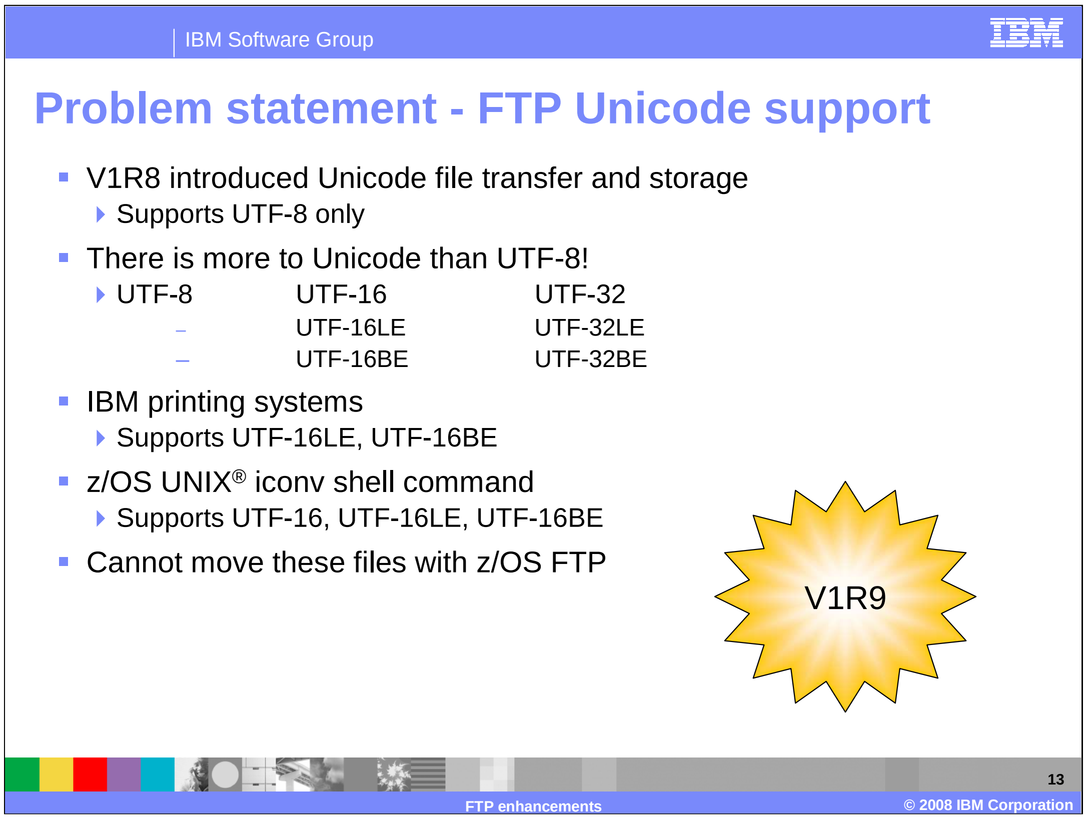
You can set UNICODEFILESYSTEMBOM to whatever you want. The setting will not affect the success or failure of the file transfer, but might impact the user of the file.



How to transfer a Unicode file

- Configure FTP with these required settings:
  - Type is ASCII
  - Mode is Stream
  - Structure File
  - Encoding is MBCS
  - MBDATACONN (UTF-8,UTF-8)
- Add this highly recommended setting at the sending host:
  - MBSENDEOL CRLF
    - Required setting if target host is z/OS
    - This is the default value
- Add these settings at the receiving host:
  - MBREQUIRELASTEOL
    - TRUE – sender is z/OS FTP
    - FALSE – sender is windows FTP client
    - Consult vendor for other platforms
  - UNICODEFILESYSTEMBOM
    - ASIS – store file with BOM only if file sent with BOM
    - ALWAYS – always store file with BOM
    - NEVER – never store file with BOM
- Transfer your file!
- The Type, Mode, and Structure settings are default values. If need be, you can reset them with the subcommands: ascii, mode stream, structure file.
- The remaining settings can be set by coding statements in FTP.DATA, or with locsite and site subcommands, or with the server SITE command.
- See IP User’s Guide and Commands for information about these subcommands: site, locsite, ascii, mode, structure.
- See IP Configuration Reference for information about these statements: ENCODING, MBDATACONN, MBSENDEOL, MBREQUIRELASTEOL, UNICODEFILESYSTEMBOM.

These notes provide a checklist for transferring a Unicode file, and resources for more information.



IBM Software Group

Problem statement - FTP Unicode support

- V1R8 introduced Unicode file transfer and storage
  - Supports UTF-8 only
- There is more to Unicode than UTF-8!

UTF-8	UTF-16	UTF-32
–	UTF-16LE	UTF-32LE
–	UTF-16BE	UTF-32BE

- IBM printing systems
  - Supports UTF-16LE, UTF-16BE
- z/OS UNIX® iconv shell command
  - Supports UTF-16, UTF-16LE, UTF-16BE
- Cannot move these files with z/OS FTP

V1R9

13

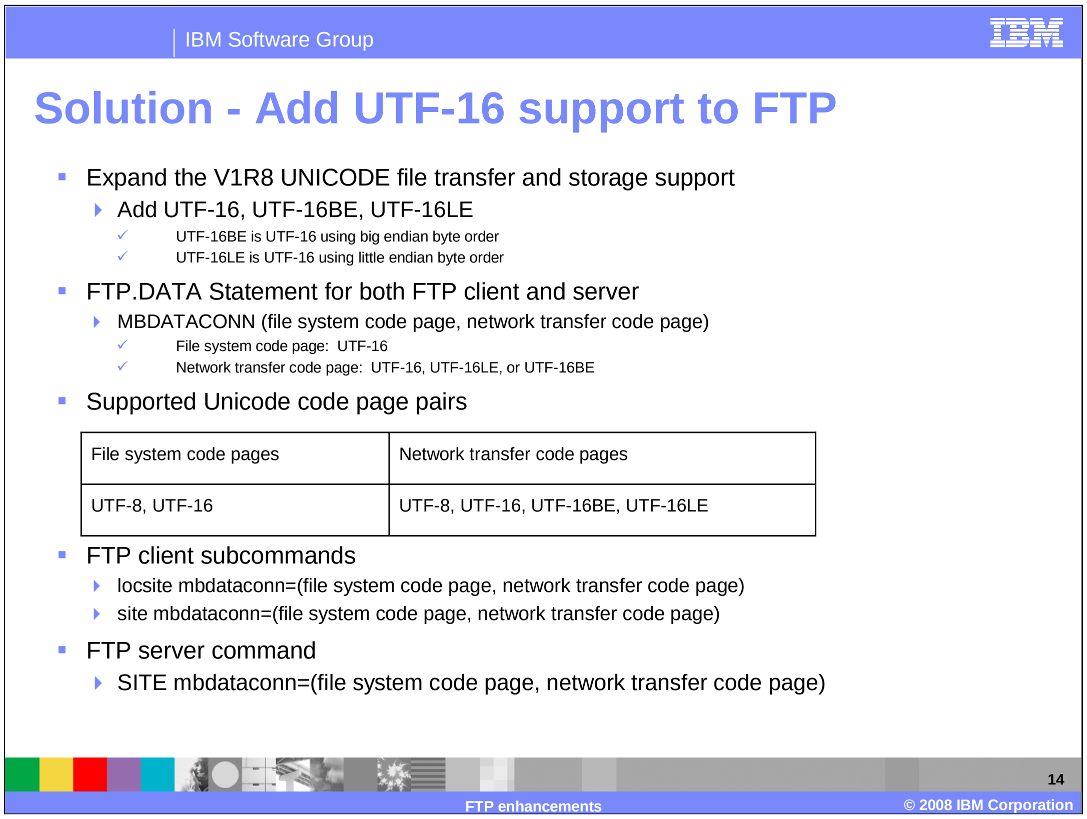
FTP enhancements

© 2008 IBM Corporation

In the z/OS V1R8 Communications Server, z/OS FTP supported Unicode file transfer and storage, but the only Unicode encoding supported was UTF-8. However, there is more to Unicode than UTF-8! The Unicode Consortium has defined the encodings listed here.

The z/OS platform has started making use of the UTF-16 class of encodings. Here you see two z/OS exploiters of UTF-16 encodings.

The problem is that you cannot move these Unicode files with z/OS FTP.



Solution - Add UTF-16 support to FTP

- Expand the V1R8 UNICODE file transfer and storage support
  - Add UTF-16, UTF-16BE, UTF-16LE
    - ✓ UTF-16BE is UTF-16 using big endian byte order
    - ✓ UTF-16LE is UTF-16 using little endian byte order
- FTP.DATA Statement for both FTP client and server
  - MBDATACONN (file system code page, network transfer code page)
    - ✓ File system code page: UTF-16
    - ✓ Network transfer code page: UTF-16, UTF-16LE, or UTF-16BE
- Supported Unicode code page pairs

File system code pages	Network transfer code pages
UTF-8, UTF-16	UTF-8, UTF-16, UTF-16BE, UTF-16LE

- FTP client subcommands
  - locsite mbdataconn=(file system code page, network transfer code page)
  - site mbdataconn=(file system code page, network transfer code page)
- FTP server command
  - SITE mbdataconn=(file system code page, network transfer code page)

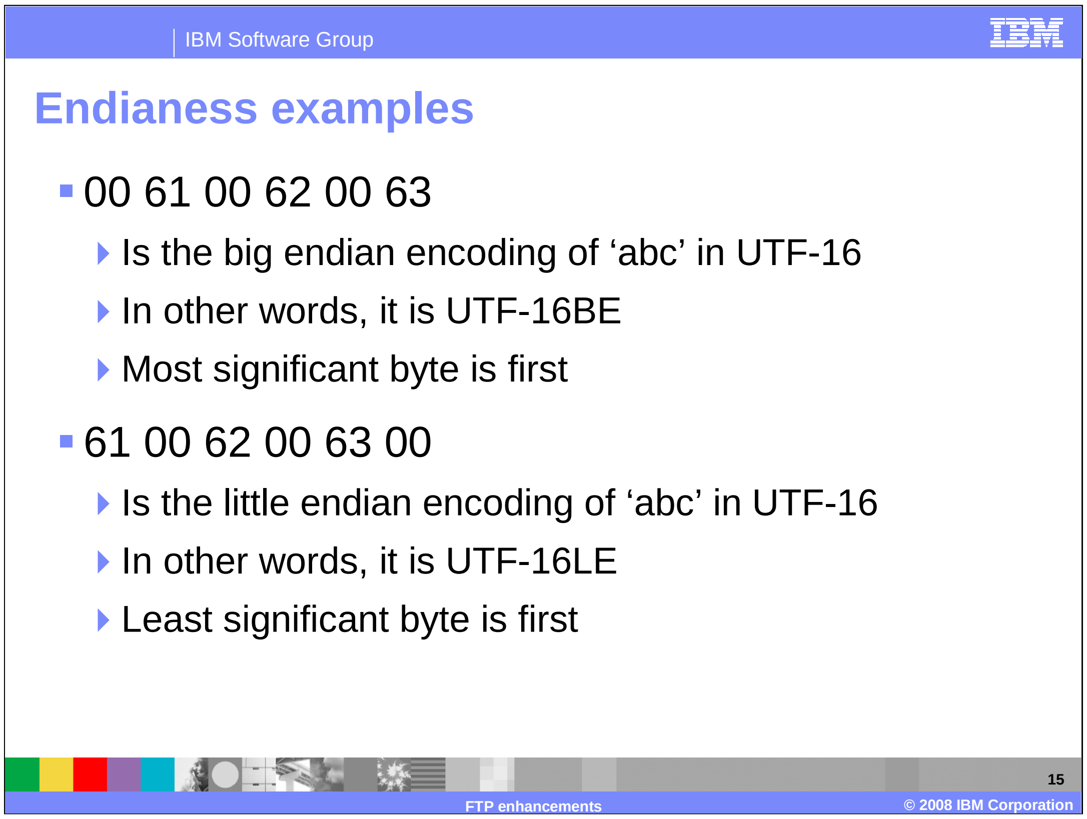
The z/OS V1R9 Communications Server builds upon the UNICODE support added in V1R8 by adding support for UTF-16. For practical purposes, UTF-16 uses two bytes per character (your notes discuss exceptions). A two byte character must use either little endian byte order or big endian byte order; therefore, UTF-16 is always either UTF-16BE or UTF-16LE. By definition, UTF-16 is UTF-16BE by default unless a BOM is present. Recall that MBDATACONN statement defines the code pages to use for multi byte file transfer, and can be specified for both the FTP client and server. For the file system code page, you can now specify UTF-16. FTP will always use UTF-16BE in the z/OS file system.

For the network transfer, you can now specify the UTF-16 encoding schemes above. FTP must be able to support either little endian or big endian encodings on the network because some platforms support only little endian encoding. FTP for z/OS will always assume that UTF-16 is equivalent to UTF-16BE (big endian UTF-16).

The chart summarizes all the MBDATACONN code page combinations supported for Unicode file transfer as of V1R9. Any choice from the **File system code pages** column can be specified with any choice from the **Network transfer code pages** column.

As well as using the FTP.DATA statement MBDATACONN, you can use locsite and site subcommands to configure the multibyte code pages. Code pages valid for the MBDATACONN FTP.DATA statement are valid for site and locsite subcommands. As a reminder, the locsite subcommand configures the FTP client; the site subcommand configures the FTP server.

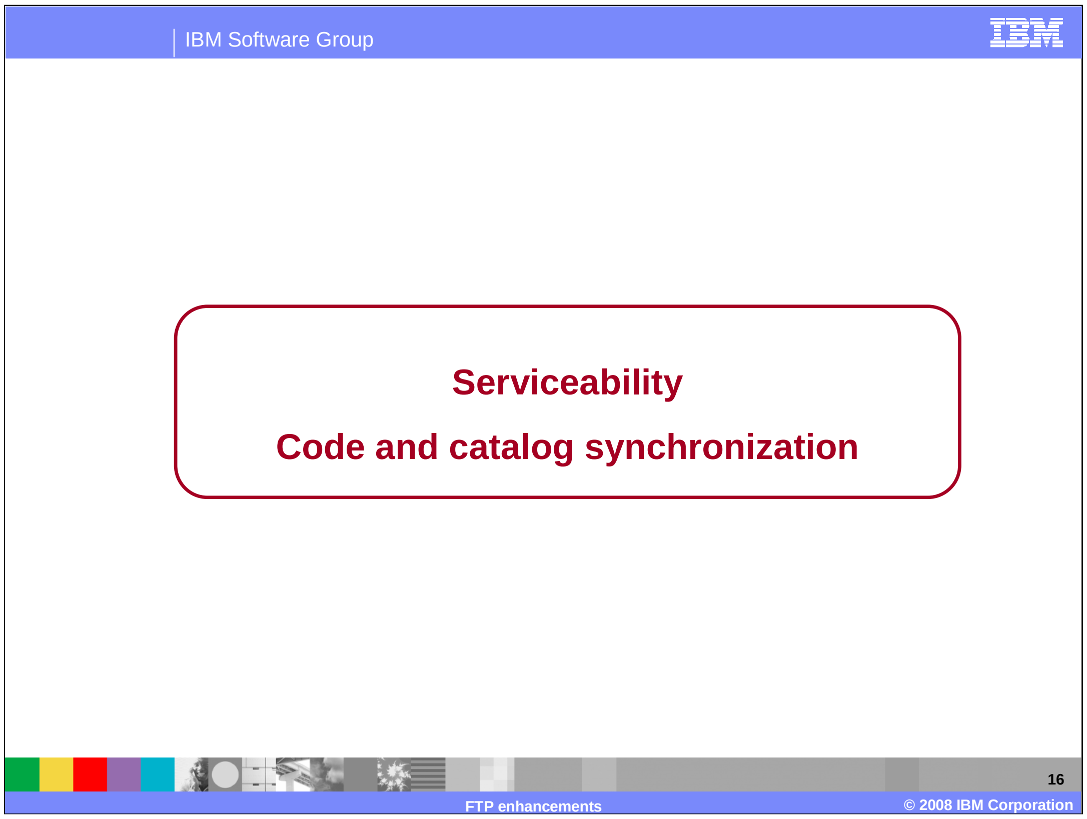
The server SITE command parameter MBDATACONN defines code pages for multibyte transfer to the server, for the current login session. The MBDATACONN parameter is enhanced to accept the new UTF-16 code pages. The z/OS client sends a SITE command to the server for you when you use the site subcommand. If you log into the server using a different FTP client such as the Windows client, you might have to use that client's QUOTE subcommand to send a SITE command to the server.



Endianess examples

- 00 61 00 62 00 63
  - Is the big endian encoding of ‘abc’ in UTF-16
  - In other words, it is UTF-16BE
  - Most significant byte is first
- 61 00 62 00 63 00
  - Is the little endian encoding of ‘abc’ in UTF-16
  - In other words, it is UTF-16LE
  - Least significant byte is first

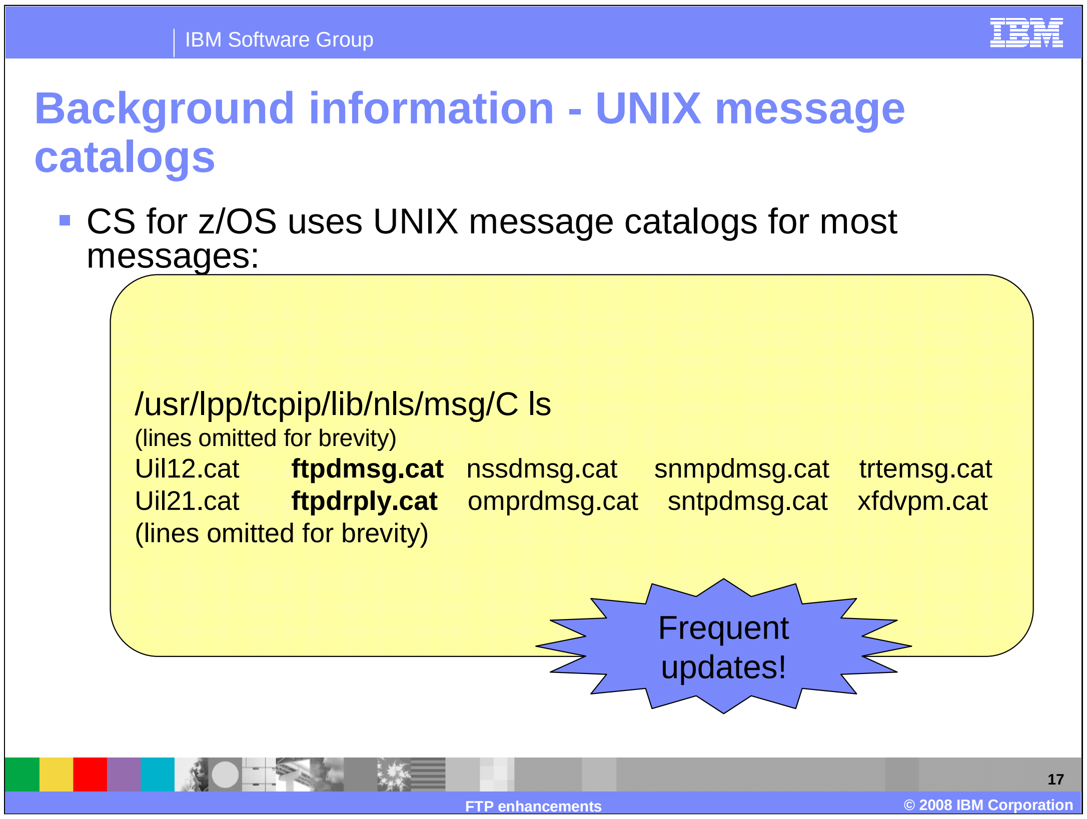
UTF-16 is always either little endian or big endian. For z/OS, UTF-16 is equivalent to UTF-16BE. These notes show the string ‘abc’ in both UTF-16 formats.



Serviceability

Code and catalog synchronization

This section describes a serviceability enhancement to FTP to verify the FTP code and FTP catalogs are synchronized.


IBM Software Group

Background information - UNIX message catalogs

- CS for z/OS uses UNIX message catalogs for most messages:

```
/usr/lpp/tcpip/lib/nls/msg/C ls
(lines omitted for brevity)
Uil12.cat     ftpdmsg.cat   nssdmsg.cat    snmpdmsg.cat   trtemsg.cat
Uil21.cat     ftpdrply.cat  omprdmsg.cat   sntpdmsg.cat   xfdvpm.cat
(lines omitted for brevity)
```

Frequent updates!

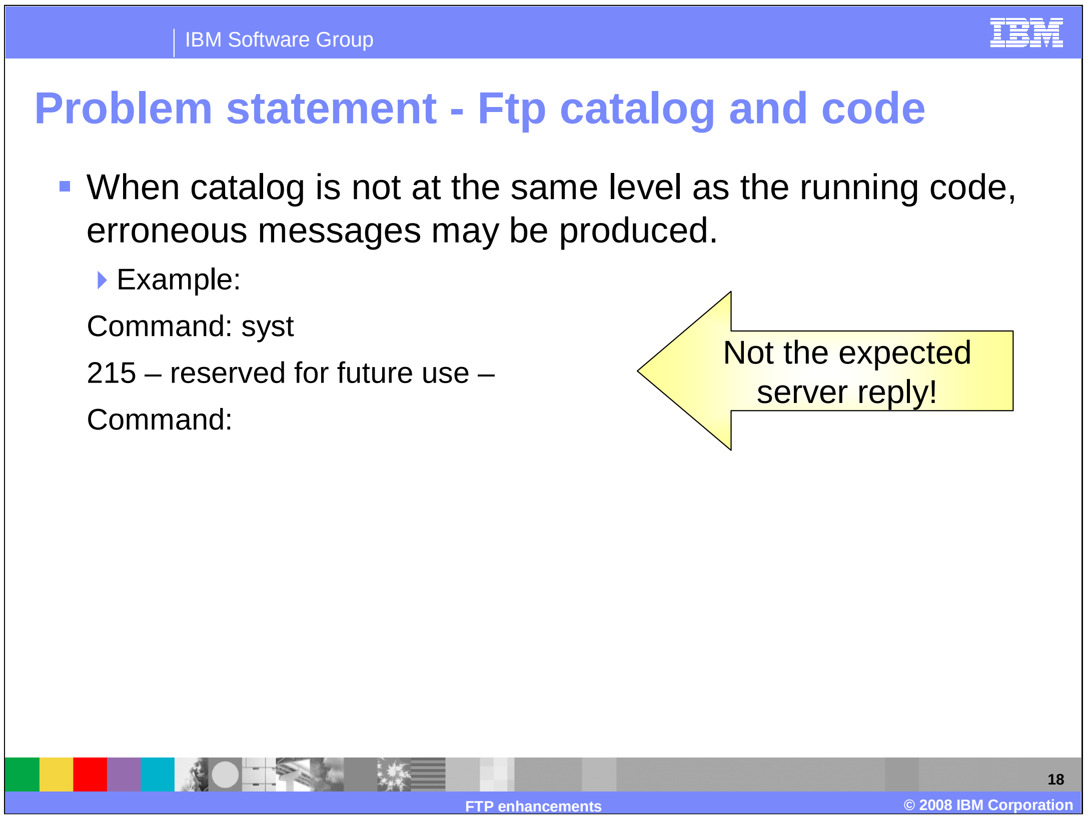
17

FTP enhancements © 2008 IBM Corporation

The directory shown above is where CS for z/OS provides the UNIX message catalogues it uses, along with a partial listing of the UNIX message catalogues that CS for z/OS uses. Files of with extension .cat are formatted message files, the executable form of the message catalogue. For this line item, note the message catalogues used by FTP, highlighted in **bold** font.

The .cat files are not message source you can translate, but some customers are reverse engineering the catalogue source from the .cat files. Although the messages in a message catalogue can be changed without affecting the source code, you cannot change the order or number of messages in the catalogue. The order and number of messages is strictly bound to the running code. The next page lists restrictions on what you can change in a message catalogue; for this line item you can focus on the problem of adding messages.

FTP is constantly adding messages to its catalogues. For every release, and often for maintenance fixes as well, CS for z/OS ships FTP catalogue updates and executable code updates.



Problem statement - Ftp catalog and code

- When catalog is not at the same level as the running code, erroneous messages may be produced.
  - Example:

Command: syst

215 – reserved for future use –

Command:

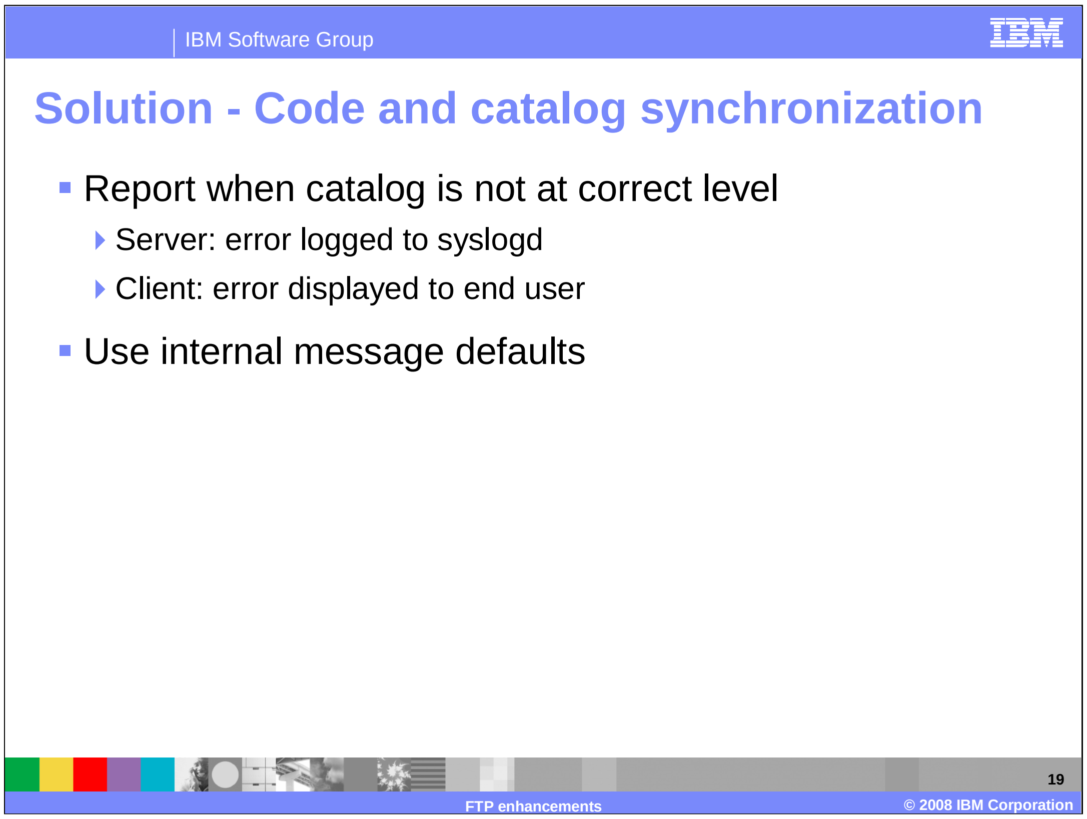
Not the expected server reply!

FTP uses the aforementioned UNIX message catalogs to supply the text for FTP end user messages and replies, and operator messages.

If these catalogs are not at the same level as the executing code, the wrong text could be fetched from the catalog. For example, if the catalog is down level and FTP is retrieving a new message, the new message may not be defined in the catalog. A message such as '-- reserved for future use –' might be displayed, as in this example. More seriously, had the reply code been 215- instead of 215 – (a space separates 5 and -), the client would, as specified by RFC 959, wait for another 215 reply. Thus, the client would hang – the server won't be sending another reply 215!
Finally, the reply code could be completely different.

A catalog synchronization error can arise from several sources:
 The UNIX file system containing the catalogs may be incorrectly mounted
 The customer has applied code updates but omitted to apply a catalog update
 - The customer has changed the catalog and it is not kept current when new maintenance is applied.

This support is intended to notify the customer when either of the FTP catalogs are not synchronized with the executing code.


IBM Software Group

Solution - Code and catalog synchronization

- Report when catalog is not at correct level
  - Server: error logged to syslogd
  - Client: error displayed to end user
- Use internal message defaults

19

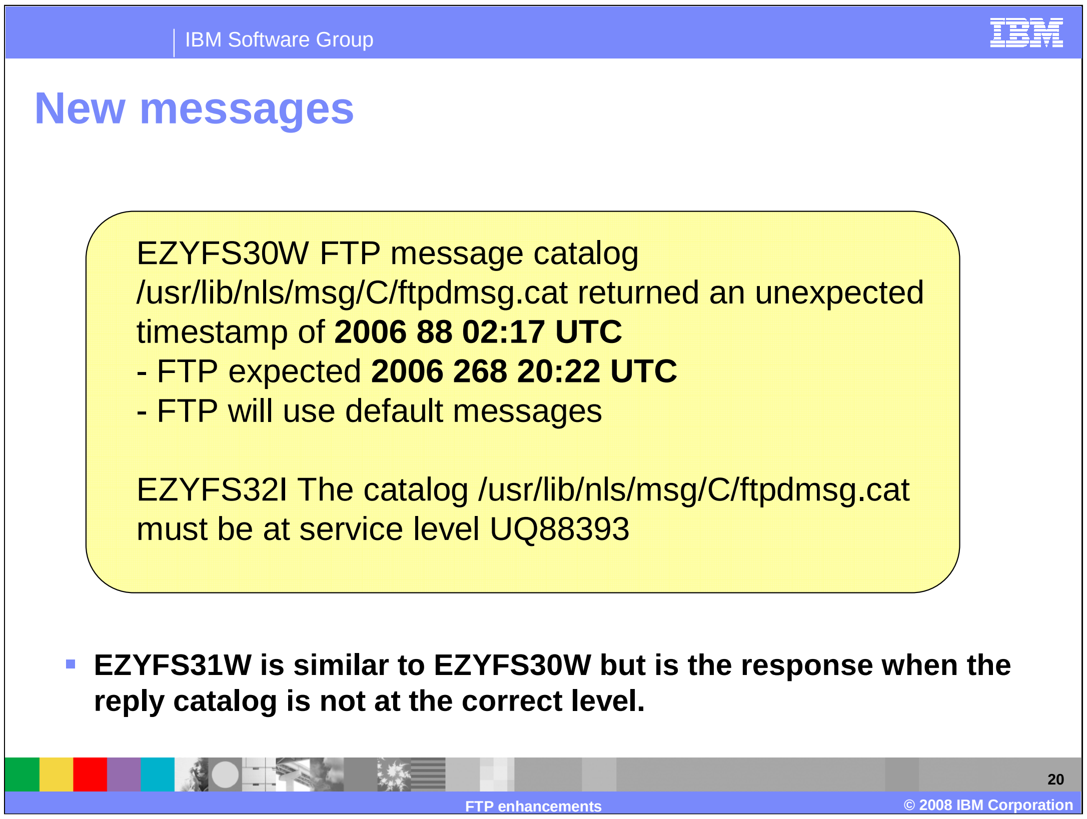
FTP enhancements

© 2008 IBM Corporation

To solve this, FTP takes two actions when the catalog is detected at an incorrect level.

A message describing the code and catalog mismatch is generated with information enabling the correct identification of the required message catalog level.

The catalog is closed and default messages are used.


IBM Software Group

New messages

EZYFS30W FTP message catalog /usr/lib/nls/msg/C/ftpdmsg.cat returned an unexpected timestamp of **2006 88 02:17 UTC**
- FTP expected **2006 268 20:22 UTC**
- FTP will use default messages

EZYFS32I The catalog /usr/lib/nls/msg/C/ftpdmsg.cat must be at service level UQ88393

- **EZYFS31W is similar to EZYFS30W but is the response when the reply catalog is not at the correct level.**

20

FTP enhancements

© 2008 IBM Corporation

The message (EZYFS30W for the message catalog and EZYFS31W for the reply catalog) identifies the name of the catalog that FTP is connected to and the unexpected time stamp extracted from the message catalog.

In this example:

- the time stamp is '**2006 88 02:17**' (that is, March 29, 2006 at 2:17 am).

- the code was expecting **2006 268 20:22** (that is, Sept 25, 2006 at 8:22 pm).

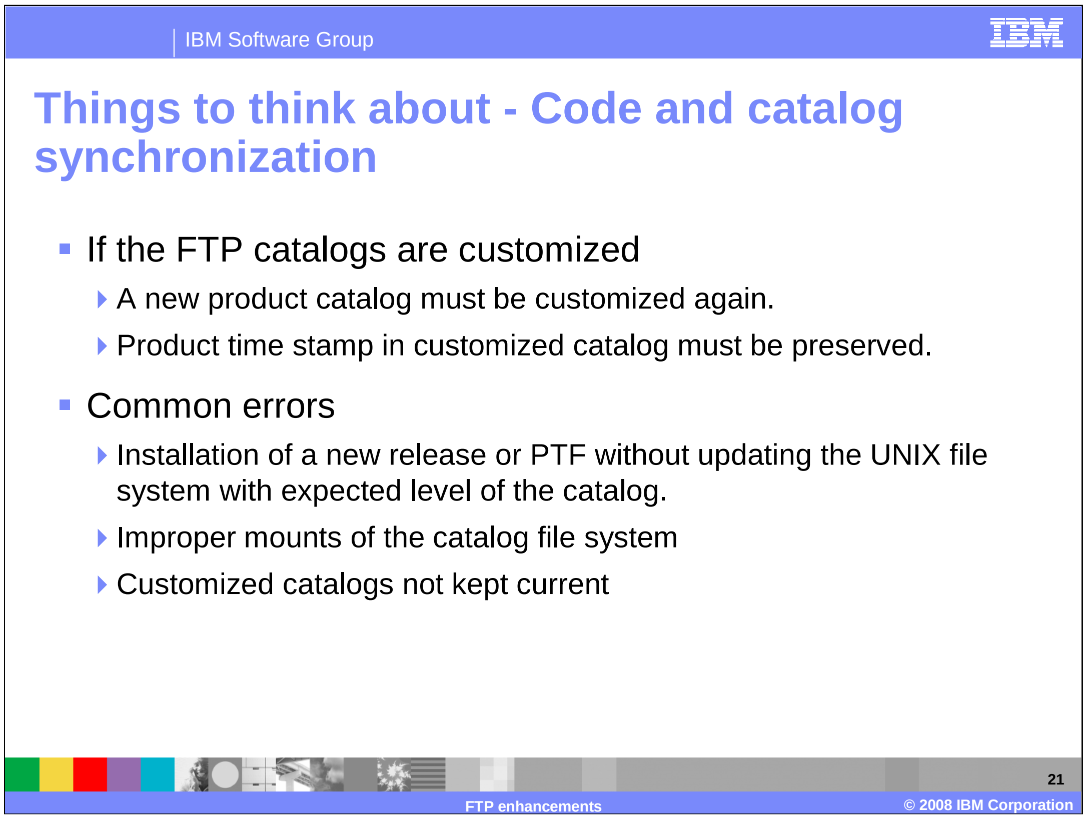
A date and time stamp are contained within the catalog and indicates when the catalog was built for distribution. The expected time stamp is built into the executing code when it is packaged. For a catalog to be considered synchronized, these time stamps must be identical.

Also, a reminder is issued that default messages will be used.

Message EZYFS32I identifies the dataset name of the catalog, and the expected service level.

In this case, UQ88393 is a PTF that needs to be installed to bring the catalog to the same level that the code expects.

This service level is built into the FTP code when it ships.



Things to think about - Code and catalog synchronization

- If the FTP catalogs are customized
  - A new product catalog must be customized again.
  - Product time stamp in customized catalog must be preserved.
- Common errors
  - Installation of a new release or PTF without updating the UNIX file system with expected level of the catalog.
  - Improper mounts of the catalog file system
  - Customized catalogs not kept current

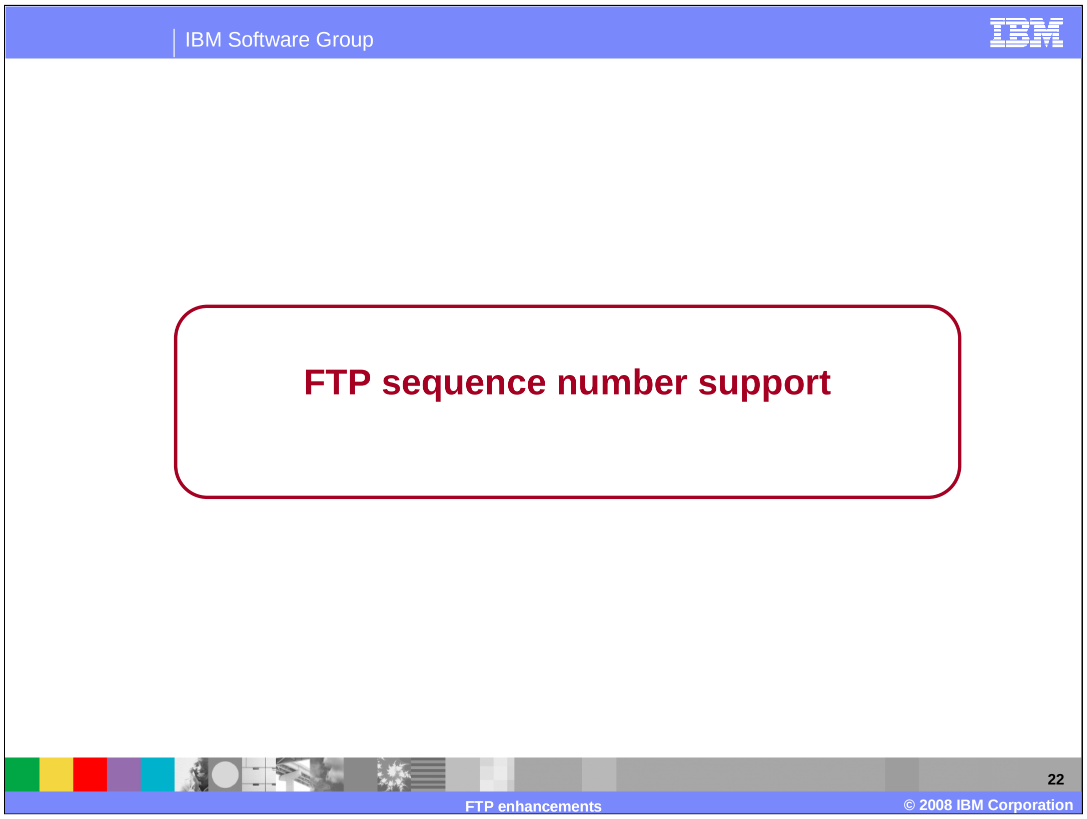
When IBM updates an FTP catalog that you have customized, the new product catalog will need to be updated with your local customizations. Along with these updates, the time stamp in the product catalog must be preserved. See the z/OS Communications Server: IP Configuration Guide topic ,“ Customizing FTP Message Catalogs“ for information on preserving the time stamp.

Catalogs and code can lose synchronization when migrating to a new release or when new product updates are applied.

This is typically caused by mounting an incorrect UNIX File System or not updating the UNIX File System to contain the correct level of the message catalog.

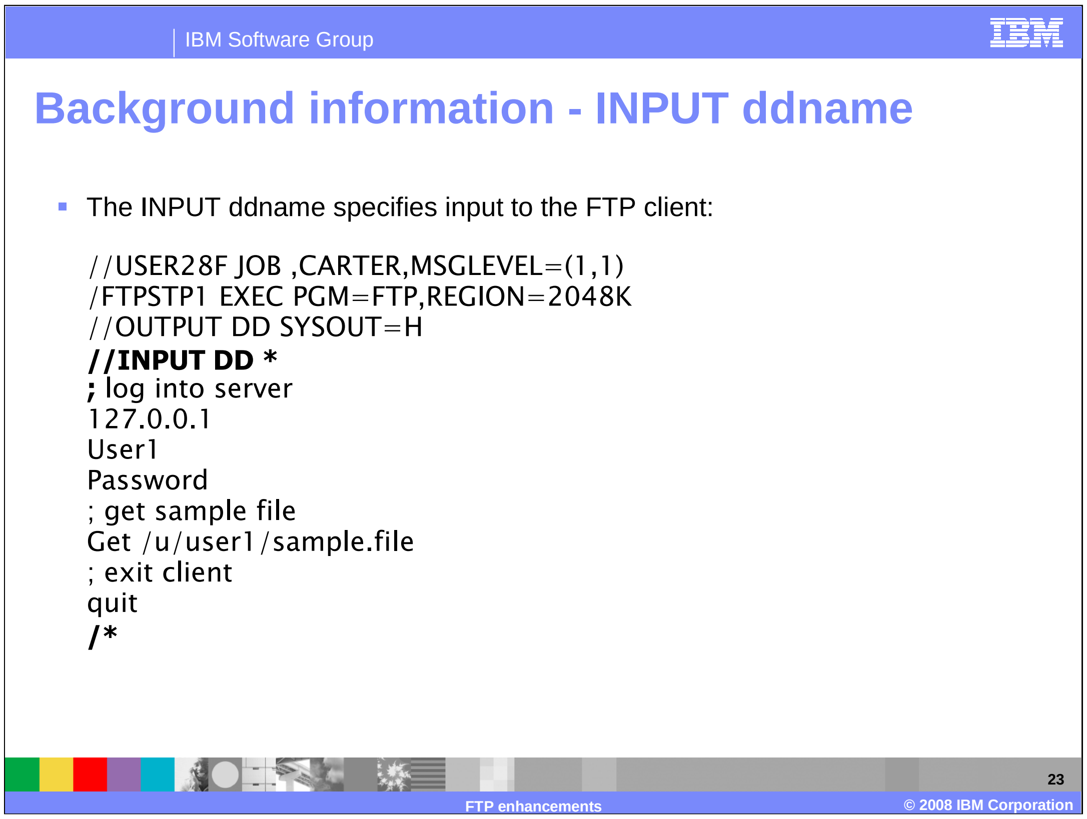
If a customized catalog is being used, it must be kept current with any maintenance that is applied.

If the catalog is not kept current with the proper time stamp, the time stamp in the local catalog will not match the time stamp in the product catalog. FTP will detect this and use internal message defaults and any customized messages will not be used.



FTP sequence number support

This section describes the new support to detect and optionally remove sequence numbers in a batch job.



Background information - INPUT ddname

- The INPUT ddname specifies input to the FTP client:

```
//USER28F JOB ,CARTER,MSGLEVEL=(1,1)
/FTPSTP1 EXEC PGM=FTP,REGION=2048K
//OUTPUT DD SYSOUT=H
//INPUT DD *
; log into server
127.0.0.1
User1
Password
; get sample file
Get /u/user1/sample.file
; exit client
quit
/*
```

When the FTP client is invoked from JCL, the ddname INPUT describes where the responses to the FTP client's prompts are.

The ddname INPUT could point to a data set, a file, or a set of concatenated files. The contents could be defined in the JCL itself, as in this example.

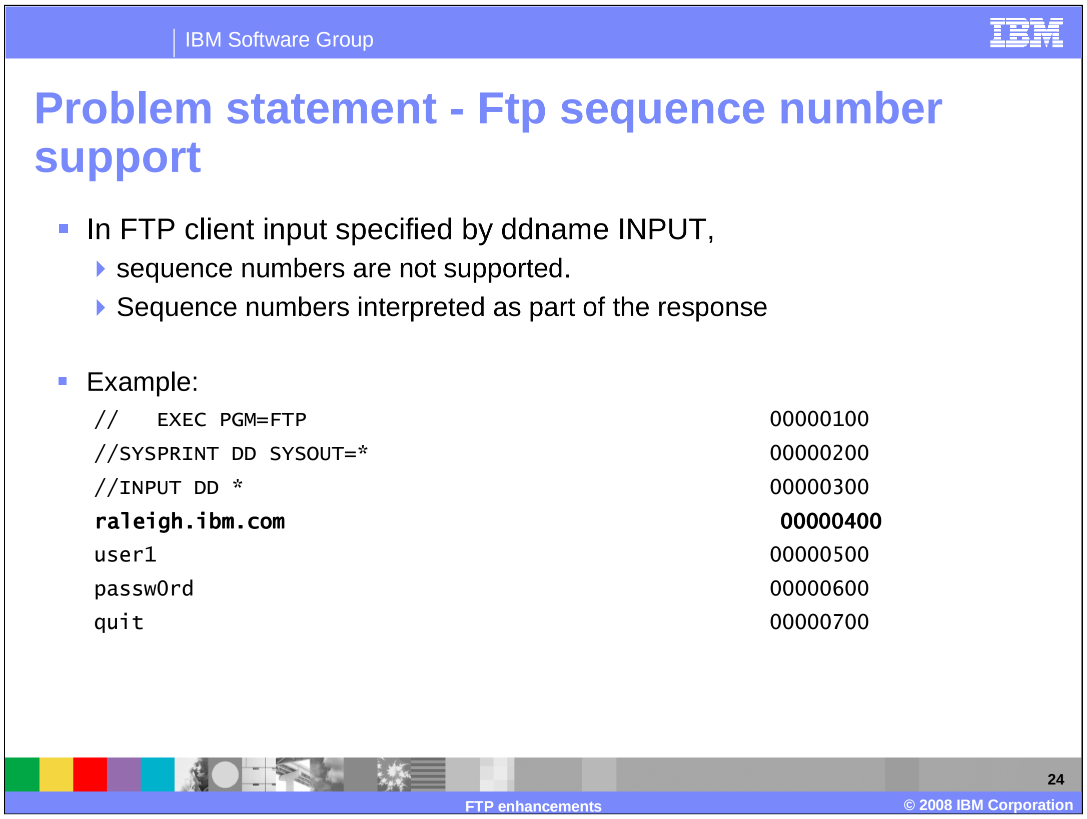
The FTP client supports the use of comments in the input stream when the client is invoked this way. In this example, the lines

; log into server

; get sample file

; exit client

are comments, because the first column is a semi-colon ';'. The FTP client detects the semi-colon and discards the rest of the line.



Problem statement - Ftp sequence number support

- In FTP client input specified by ddname INPUT,
  - sequence numbers are not supported.
  - Sequence numbers interpreted as part of the response
- Example:

```
//   EXEC PGM=FTP                                   00000100
//SYSPRINT DD SYSOUT=*                              00000200
//INPUT DD *                                        00000300
raleigh.ibm.com                                      00000400
user1                                               00000500
passw0rd                                            00000600
quit                                                00000700
```

In batch, the FTP client reads input using the //INPUT DD statement which contains a list of the responses to each prompt by the FTP client.

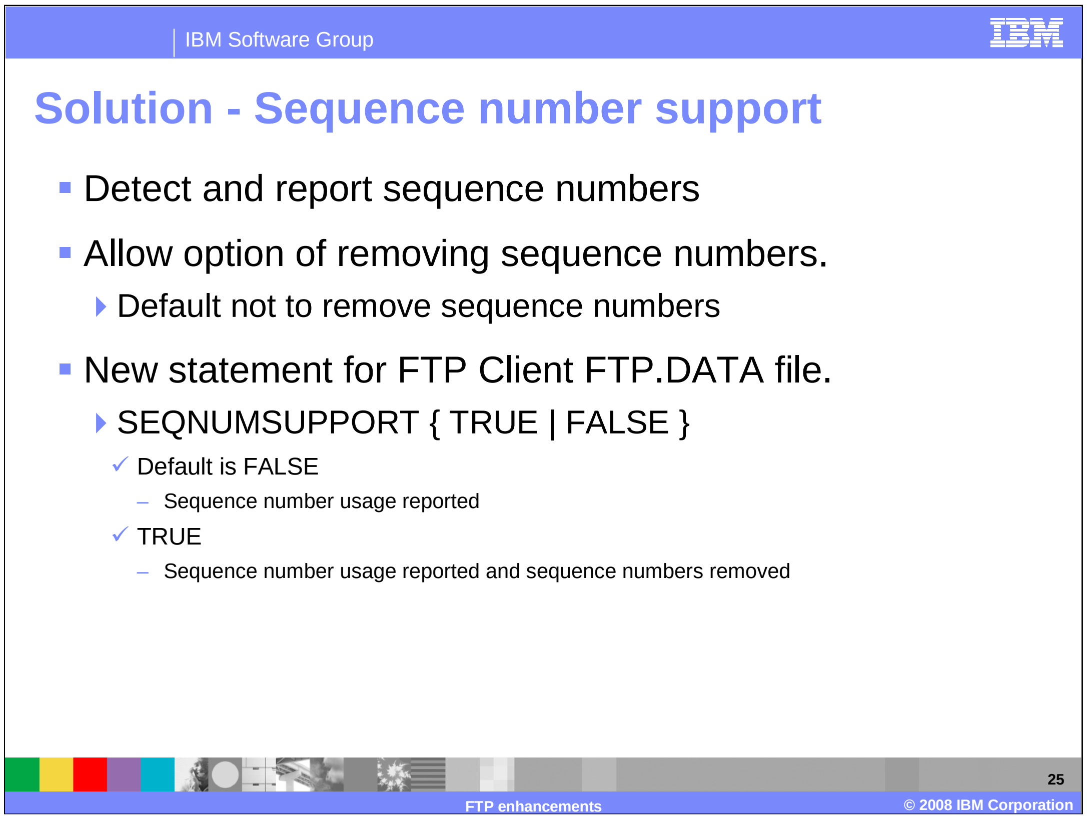
When updating this file, the user may accidentally activate sequence number support and each line of the input file becomes sequence numbered.

As most editors display line numbers, it is not obvious that the file actually may contain sequence numbers.

As FTP does not support sequence numbers, when the command input is read, FTP will interpret the sequence numbers as part of the FTP command.

This typically results in a command failure. As most of these errors occur in batch jobs, the problem may not immediately be noticed, causing customers delay in accomplishing their task.

At times, this has even resulted in calls to IBM because FTP is attempting to connect to an incorrect port. This is usually caused when the remote host name is part of the command input and the sequence number becomes interpreted as the remote port to which FTP is connecting. In the example, when FTP is started it will attempt to connect to PORT 400 because of the sequence number in the input stream.



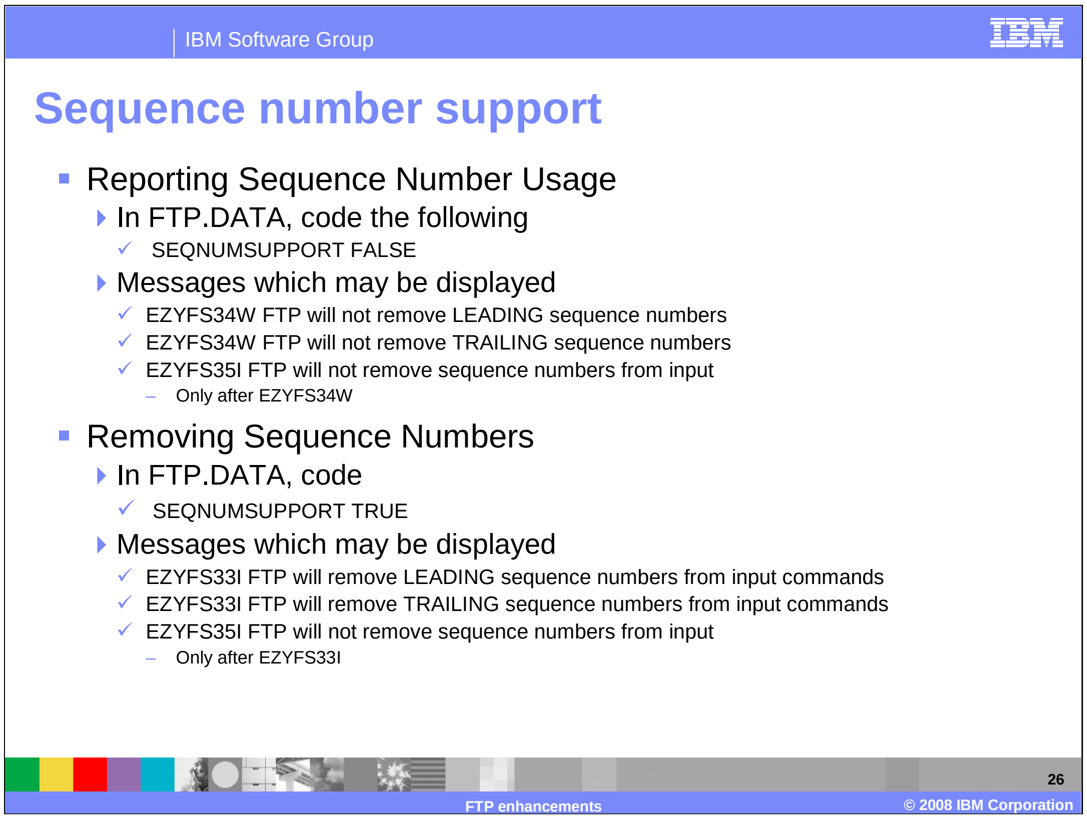
Solution - Sequence number support

- Detect and report sequence numbers
- Allow option of removing sequence numbers.
  - Default not to remove sequence numbers
- New statement for FTP Client FTP.DATA file.
  - SEQNUMSUPPORT { TRUE | FALSE }
    - ✓ Default is FALSE
      - Sequence number usage reported
    - ✓ TRUE
      - Sequence number usage reported and sequence numbers removed

The solution is to report when sequence numbers are detected in the input stream. This may still result in FTP subcommands failing, but messages will inform the user as to the probable cause. This enables FTP to function as it does with prior releases.

As an option, allow the customer to remove sequence numbers when detected. This will allow a job that would otherwise fail, to successfully complete.

A new statement is added to the FTP Client's FTP.DATA file. The keyword is SEQNUMSUPPORT and when coded as FALSE, sequence number usage will be reported but not removed. When SEQNUMSUPPORT TRUE is coded, sequence numbers will not only be reported, they will be removed.

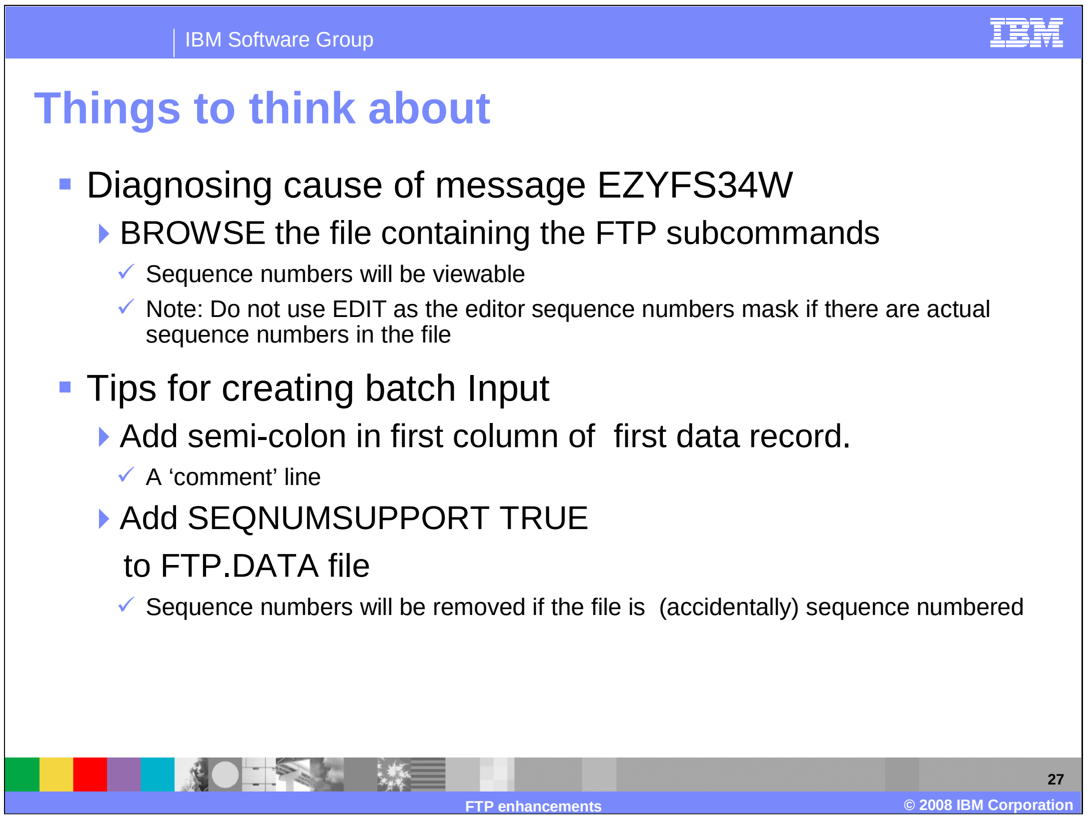


Sequence number support

- Reporting Sequence Number Usage
  - In FTP.DATA, code the following
    - ✓ SEQNUMSUPPORT FALSE
  - Messages which may be displayed
    - ✓ EZYFS34W FTP will not remove LEADING sequence numbers
    - ✓ EZYFS34W FTP will not remove TRAILING sequence numbers
    - ✓ EZYFS35I FTP will not remove sequence numbers from input
      - Only after EZYFS34W
- Removing Sequence Numbers
  - In FTP.DATA, code
    - ✓ SEQNUMSUPPORT TRUE
  - Messages which may be displayed
    - ✓ EZYFS33I FTP will remove LEADING sequence numbers from input commands
    - ✓ EZYFS33I FTP will remove TRAILING sequence numbers from input commands
    - ✓ EZYFS35I FTP will not remove sequence numbers from input
      - Only after EZYFS33I

With SEQNUMSUPPORT FALSE coded in the FTP.DATA file, the FTP client checks if sequence numbers are present. The type of sequence number is determined by the first record read or whenever a semi-colon is detected in the first data column. If FTP sequence numbers are present, an EZYFS34W message is issued to inform the user that sequence numbers are present and the type of sequence number detected. No sequence numbers will be removed. LEADING sequence numbers typically occur when input is read from a variable length file. TRAILING sequence numbers occur when input is read from fixed length file. Message EZYFS35I is issued when FTP detects a transition from processing LEADING or TRAILING sequence numbers to processing no sequence numbers.

With SEQNUMSUPPORT TRUE coded in the FTP.DATA, FTP will detect and remove sequence numbers. When FTP transitions from one type of sequence number processing to another, it will output EZYFS33I or EZYFS35I. EZYFS33I is issued when FTP will be removing LEADING or TRAILING sequence numbers. When EZYFS35I is output, it means that a previous EZYFS33I message was output indicating the type of sequence numbers FTP was removing. However, the EZYFS35I message indicates that FTP will no longer remove sequence numbers from the input.



Things to think about

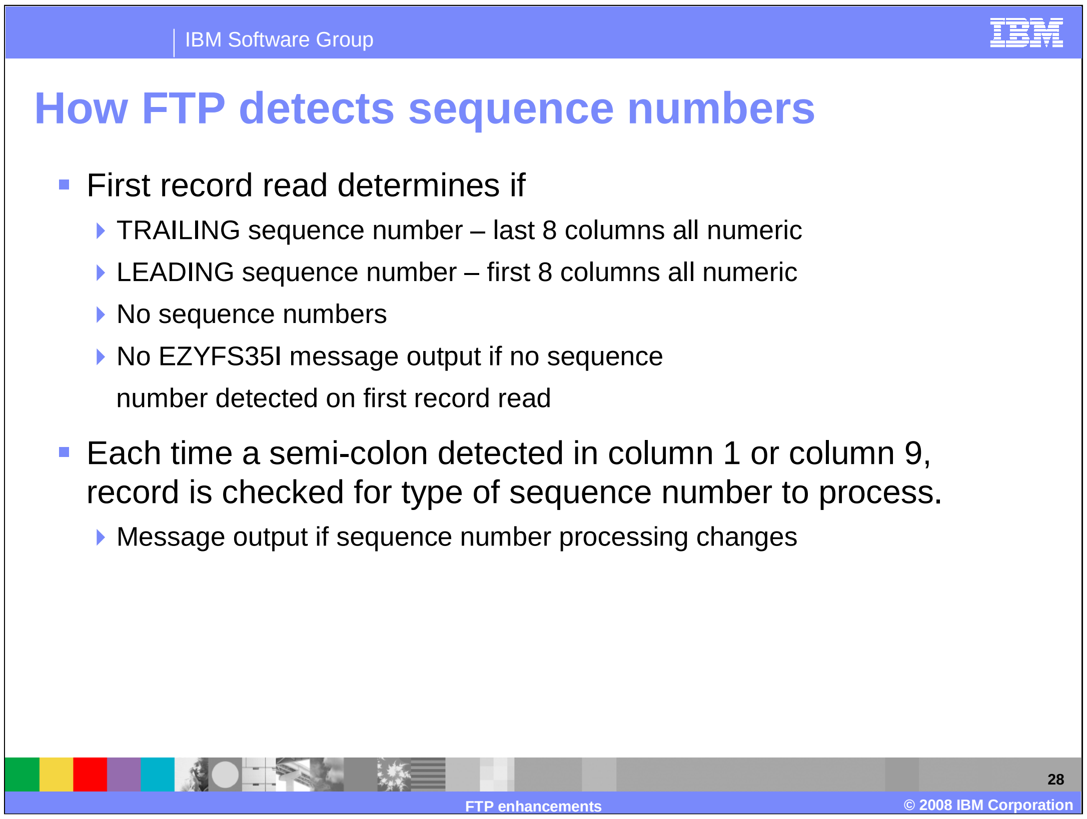
- Diagnosing cause of message EZYFS34W
  - BROWSE the file containing the FTP subcommands
    - ✓ Sequence numbers will be viewable
    - ✓ Note: Do not use EDIT as the editor sequence numbers mask if there are actual sequence numbers in the file
- Tips for creating batch Input
  - Add semi-colon in first column of first data record.
    - ✓ A 'comment' line
  - Add SEQNUMSUPPORT TRUE

    to FTP.DATA file
    - ✓ Sequence numbers will be removed if the file is (accidentally) sequence numbered

If an FTP job fails and a EZYFS34W message has been issued before the subcommand fails, failure could be the result of sequence numbers in the file. BROWSE the file, including any concatenated files, to determine if they contain sequence numbers or not. Use BROWSE because most editors will always provide pseudo line numbers which mask sequence numbers.

When the FTP client reads its subcommands from the INPUT DD statement, it is not aware the subcommands are being input from more than one file. To insure that FTP uses the proper sequence numbering scheme, add a semi-colon as the first data column of the first record of each file. This semi-colon protects the file from any accidental sequence numbering if SEQNUMSUPPORT TRUE is coded in the FTP.DATA file. If a conflicting sequence number is detected, EZYFS34W will be issued the first time a mismatched sequence number is detected.

For example, suppose that the original file starts off with no sequence numbers and then a concatenated file is read in that has TRAILING sequence numbers. If this second file has the semi-colon coded in the first record, FTP issues an "EZYFS33I FTP will remove TRAILING sequence numbers from input commands" message. This enables the commands within this second file to be processed successfully.



How FTP detects sequence numbers

- First record read determines if
  - TRAILING sequence number – last 8 columns all numeric
  - LEADING sequence number – first 8 columns all numeric
  - No sequence numbers
  - No EZYFS35I message output if no sequence number detected on first record read
- Each time a semi-colon detected in column 1 or column 9, record is checked for type of sequence number to process.
  - Message output if sequence number processing changes

When FTP reads the first record, it determines the type of sequence number by examining the last 8 columns and then the first eight columns of data to determine if it is numeric.

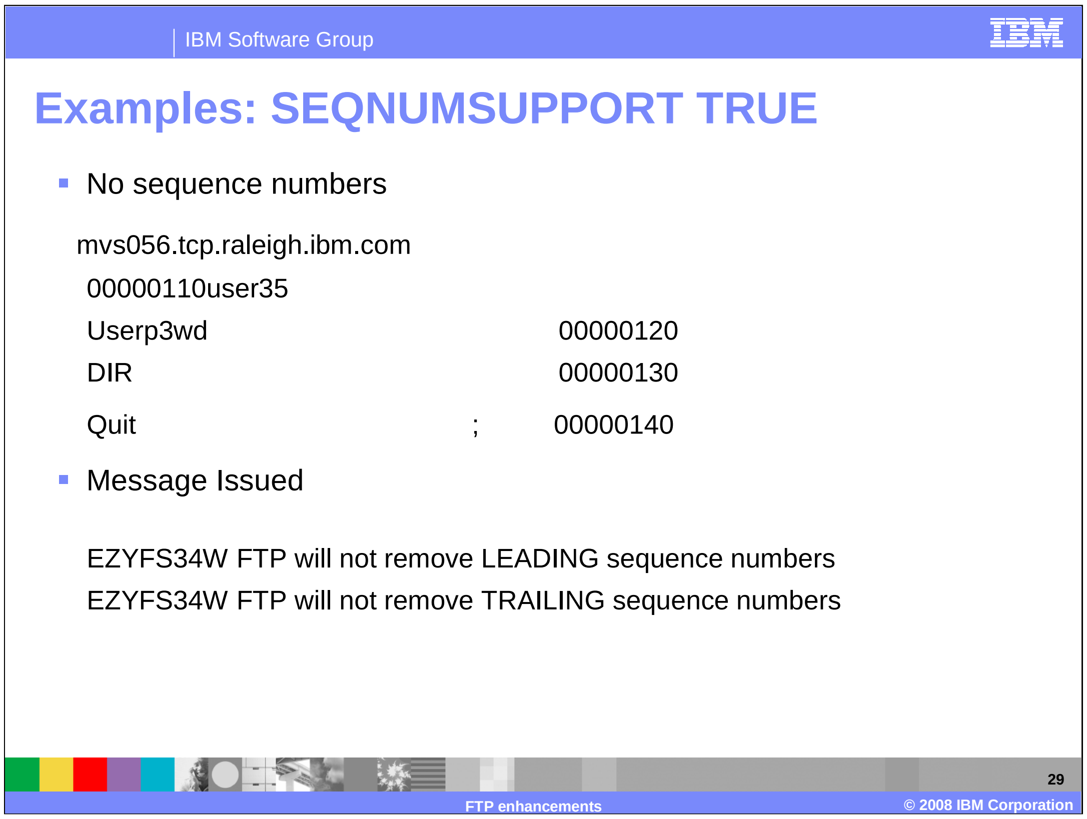
TRAILING sequence numbers have numerals in the last 8 columns which will be replaced with blanks

LEADING sequence numbers have numerals in the first 8 columns and will have the data shifted left 8 columns, with the last 8 columns replaced with blanks.

With fixed length records the first data column is 1. With variable length records, the first data column is 9, because the first 8 columns are occupied by the sequence number.

If a semi-colon is detected in these columns, FTP re-evaluates the type of sequence numbers it expects to process.

When processing TRAILING or LEADING sequence numbers and the expected sequence number does not appear in the columns expected, data will not be removed and command will be processed as entered.



Examples: SEQNUMSUPPORT TRUE

- No sequence numbers

```
mvs056.tcp.raleigh.ibm.com
00000110user35
Userp3wd                              00000120
DIR                                   00000130
Quit                           ;      00000140
```

- Message Issued

```
EZYFS34W FTP will not remove LEADING sequence numbers
EZYFS34W FTP will not remove TRAILING sequence numbers
```

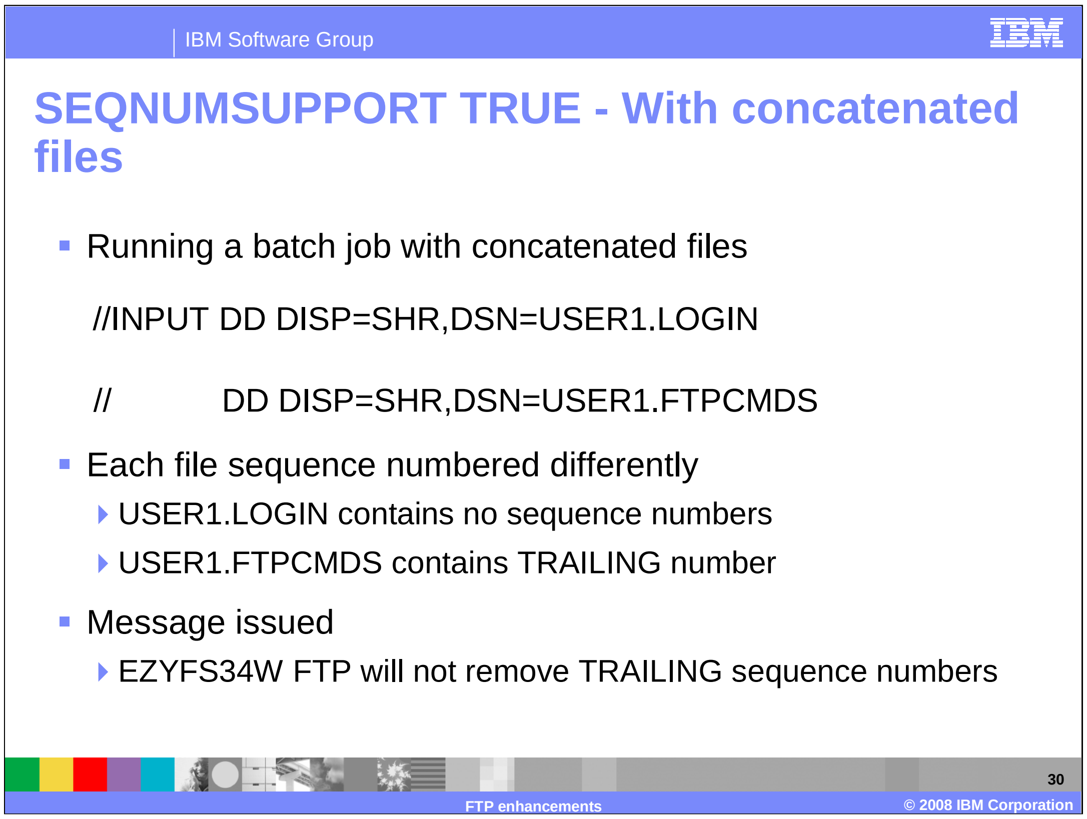
This is an example of a file which contains some records with and without sequence numbers. This is likely to happen when the initial file is created without sequence numbers and then updated with an editor which adds sequence numbers.

The first record of the file contains no sequence numbers. Any sequence numbers in the file will not be stripped off. FTP does not issue any message in this case because there is no action that FTP will be taking. This makes this support transparent with previous releases.

When the second line is read, FTP will detect the sequence number because columns 1 through 8 are numeric. This is in conflict with the original sequence number detected on the first record which indicated the file contains no sequence numbers. Message ‘EZAFS34W message is issued to indicate LEADING sequence numbers will not be removed.

When the third line is read, the last 8 columns contain numerals. Message EZAFS34W is issued again indicating that TRAILING sequence numbers will not be removed.

Each of these messages will only be issued one time to give an indication of why the subcommand may fail. Thus, no additional message will be issued when the fourth line is read.



SEQNUMSUPPORT TRUE - With concatenated files

- Running a batch job with concatenated files

```
//INPUT DD DISP=SHR,DSN=USER1.LOGIN

//           DD DISP=SHR,DSN=USER1.FTPCMDS
```

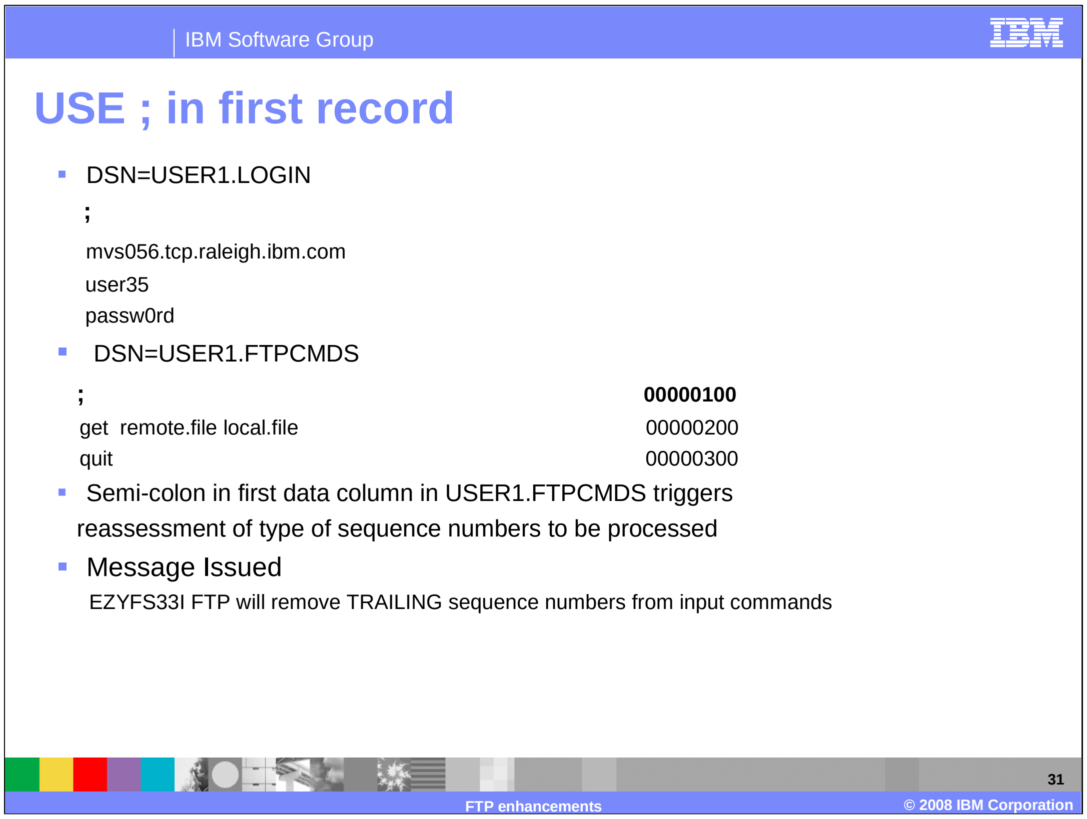
- Each file sequence numbered differently
  - USER1.LOGIN contains no sequence numbers
  - USER1.FTPCMDS contains TRAILING number
- Message issued
  - EZYFS34W FTP will not remove TRAILING sequence numbers

Sequence number support becomes more complicated when multiple files or members are concatenated as input. This example depicts two files being used as input to FTP. One file has no sequence numbers and the other file contains trailing sequence numbers.

When the USER1.LOGIN file is read, FTP detects no sequence numbers and will process all remaining commands without removing sequence numbers.

When USER1.FTPCMDS is read, FTP is not aware that data is being read from the second file. FTP reads the input as one continuous file. When the first record of the 2nd file is read, EZYFS34W is issued to indicate at least one record has been detected with an unexpected sequence number. The sequence number is not removed and can result In a command failure.

The EZYFS34W message provides a warning as to why the command may have failed.


IBM Software Group

USE ; in first record

- DSN=USER1.LOGIN

```
;
mvs056.tcp.raleigh.ibm.com
user35
passw0rd
```

- DSN=USER1.FTPCMDS

```
;                                              00000100
get  remote.file local.file                    00000200
quit                                           00000300
```

- Semi-colon in first data column in USER1.FTPCMDS triggers reassessment of type of sequence numbers to be processed
- Message Issued

```
EZYFS33I FTP will remove TRAILING sequence numbers from input commands
```

31

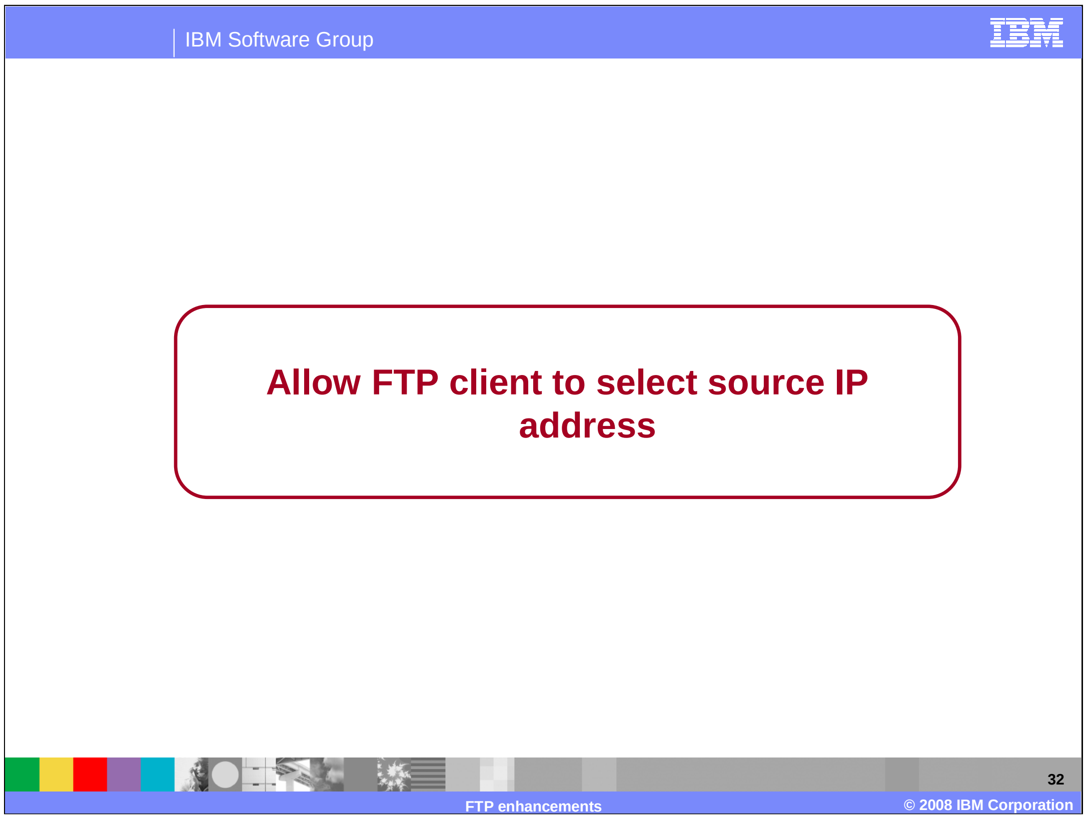
FTP enhancements © 2008 IBM Corporation

To enable the full benefit when SEQNUMSUPPORT TRUE is coded in the FTP.DATA file, add a semi-colon as the first record of any file that contains FTP commands. In the above foil, data under **DSN=USER1.LOGIN** and **DSN=USER1.FTPCMDS** show the content of the files. In this sample, the concatenation sequence results in USER1.LOGIN being read first, followed by USER1.FTPCMDS.

When the semi-colon is detected in the first data column from **DSN=USER1.LOGIN**, FTP will interrogate the line to determine the type of sequence number it contains. In this case, no sequence number is detected and it will process the file as containing no sequence numbers.

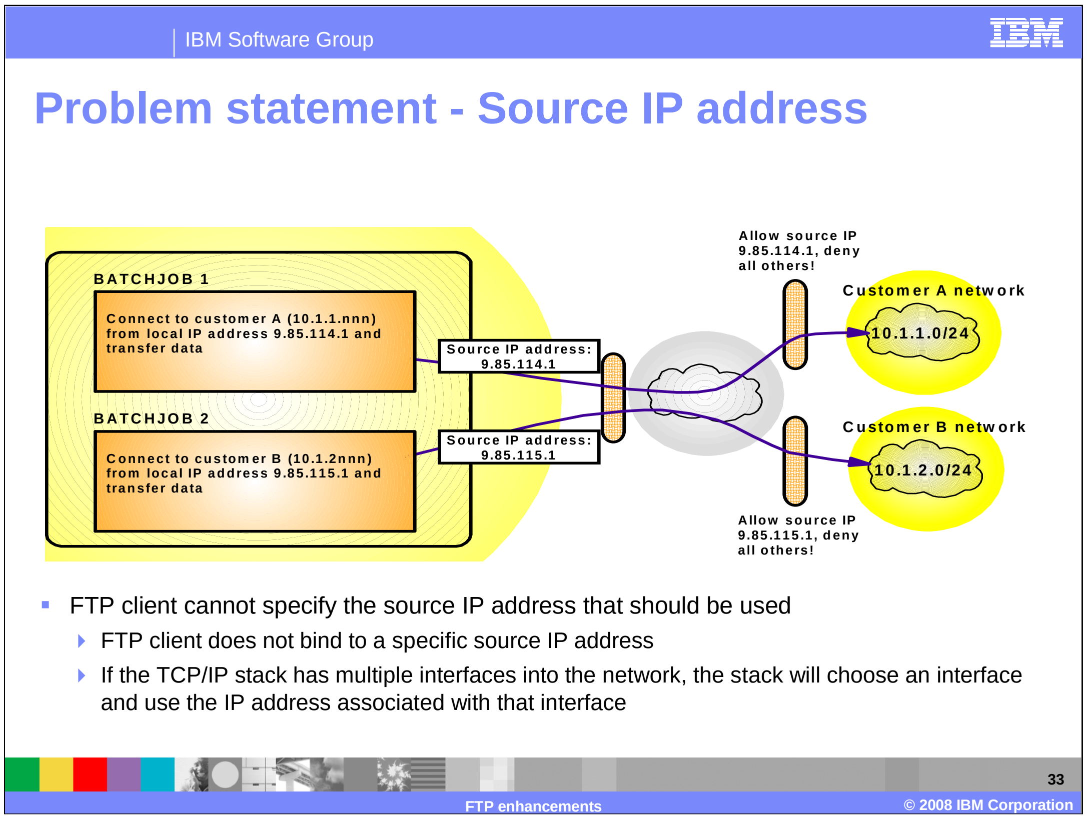
When the first line of USER1.FTPCMDS is read, it contains a semicolon which causes FTP to reassess the type of sequence number to be processed. In this case a TRAILING sequence number is detected which is different than the NO SEQUENCE numbers that FTP is currently processing. FTP will output message EZYFS33I to indicate that it will begin removing TRAILING sequence numbers.

This allows the commands in the file USER1.FTPCMDS to be processed successfully.



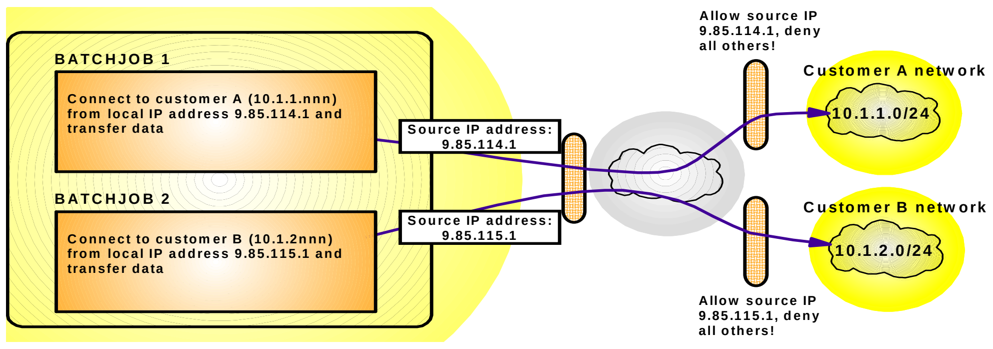
Allow FTP client to select source IP address

This section covers the new function to enables specifying the source IP address that will be used for connections from the FTP client to the FTP server.



IBM Software Group

Problem statement - Source IP address


BATCHJOB 1
Connect to customer A (10.1.1.nnn) from local IP address 9.85.114.1 and transfer data
BATCHJOB 2
Connect to customer B (10.1.2nnn) from local IP address 9.85.115.1 and transfer data
Source IP address: 9.85.114.1
Source IP address: 9.85.115.1
Allow source IP 9.85.114.1, deny all others!
Customer A network
10.1.1.0/24
Customer B network
10.1.2.0/24
Allow source IP 9.85.115.1, deny all others!

- FTP client cannot specify the source IP address that should be used
  - FTP client does not bind to a specific source IP address
  - If the TCP/IP stack has multiple interfaces into the network, the stack will choose an interface and use the IP address associated with that interface

33

FTP enhancements

© 2008 IBM Corporation

Currently, there is no way for the FTP client to specify which source IP address should be used when connecting to the FTP server.

The TCP/IP stack determines the source IP address. This can be based on TCP/IP configuration options such as Job-Specific Source IP or it may be determined when the route to the FTP server is found.

In some situations the FTP client may want to use a different source IP address when connecting to different FTP servers. In firewall configurations, it may be necessary to use a specific source IP address for the firewall to allow the connection. But, there is no way for the FTP client, itself, to specify the source IP address that should be used.

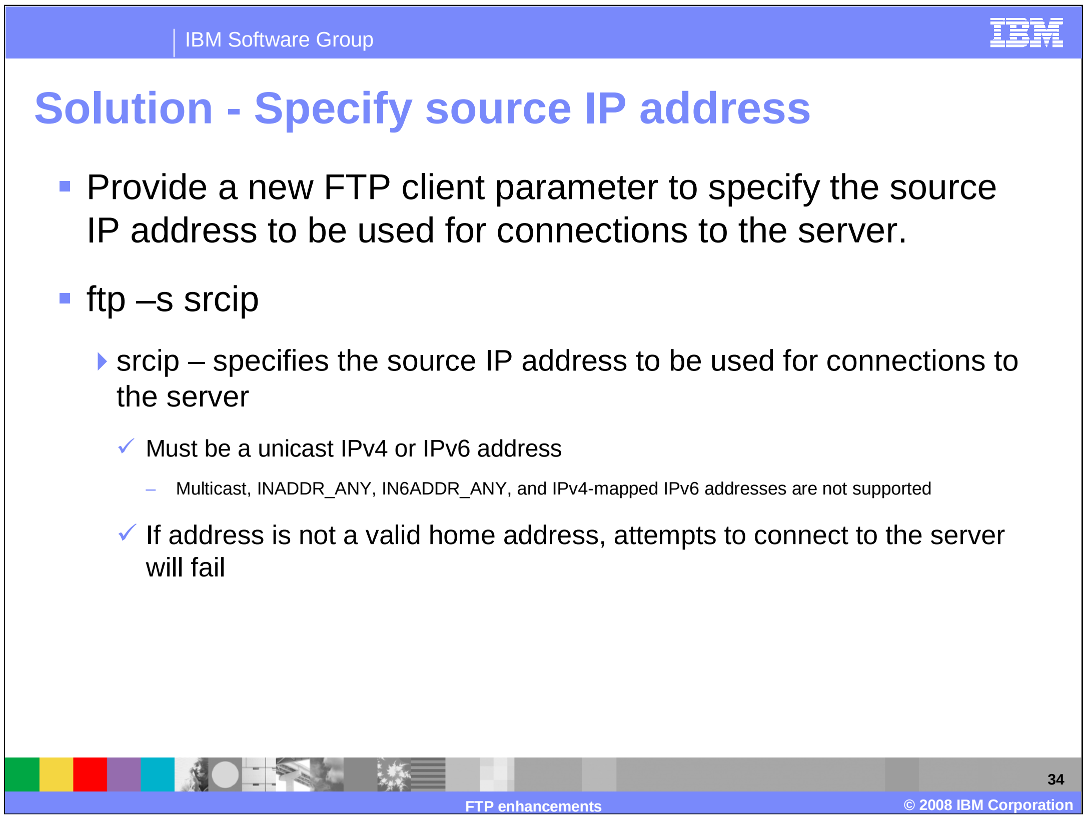
This diagram shows an example of when the FTP client may want to specify the source IP address.

In the diagram, the customer has a network setup where the z/OS system running the FTP client has two interfaces into the network.

The customer needs to be able to FTP into two other networks which are protected by firewalls. The firewalls are configured to only allow connections from specific IP addresses.

So the only way to successfully FTP into “Customer A network”, is to use a source IP address of 9.85.114.1

Since there is no way for the FTP client to specify a source IP address, there is no guarantee that the TCP/IP stack will choose the correct interface. Since there are two interfaces into the network the TCP/IP stack may choose either interface.



Solution - Specify source IP address

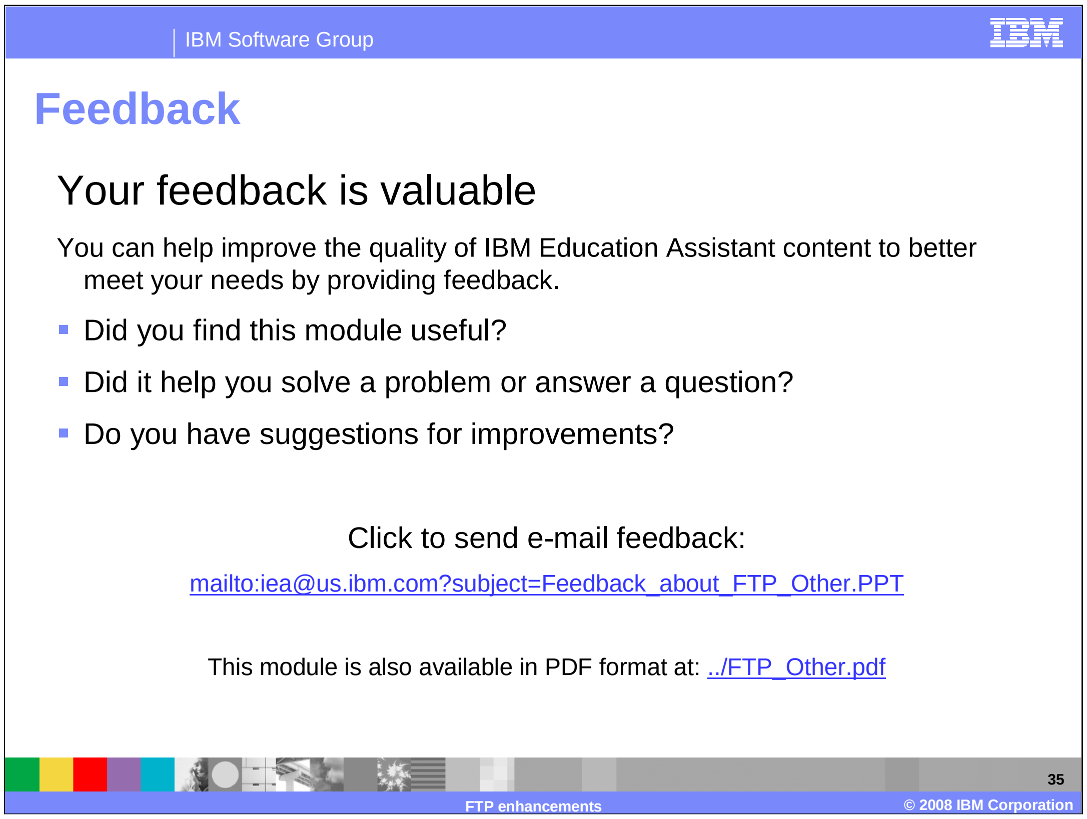
- Provide a new FTP client parameter to specify the source IP address to be used for connections to the server.
- ftp –s srcip
  - srcip – specifies the source IP address to be used for connections to the server
    - ✓ Must be a unicast IPv4 or IPv6 address
      - Multicast, INADDR_ANY, IN6ADDR_ANY, and IPv4-mapped IPv6 addresses are not supported
    - ✓ If address is not a valid home address, attempts to connect to the server will fail

A new FTP client command line parameter will be added to allow the specification of the source IP address that will be used for connections to the FTP server.

The new command line parameter is:

-s srcip

The srcip must be a unicast IPv4 or IPv6 address. Multicast , INADDR_ANY, IN6ADDR_ANY, and IPv4-mapped IPv6 addresses are not supported. If a non-valid address is specified then the FTP command will be rejected. If a valid address is specified, but the address is not an active home address on the TCP/IP stack, connections to the server will fail.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

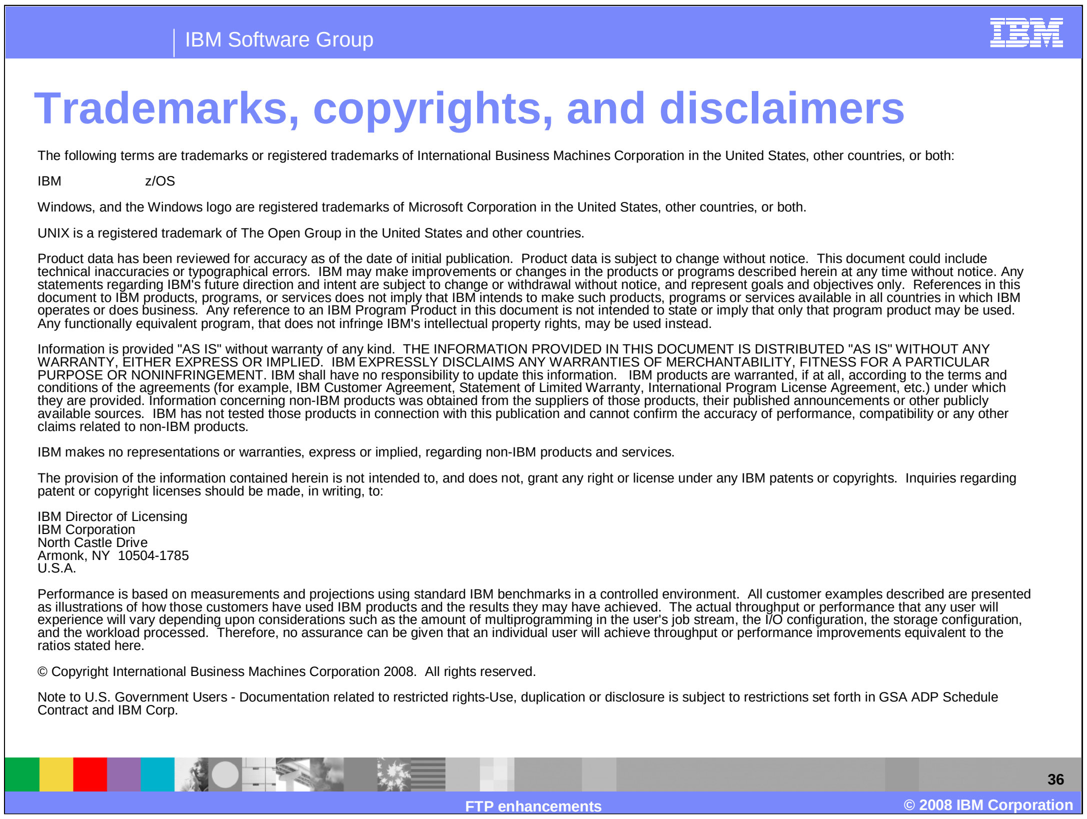
- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_FTP_Other.PPT

This module is also available in PDF format at: ../FTP_Other.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.



Trademarks, copyrights, and disclaimers