The webinar will begin shortly

SDS

Quality Mainframe Software since 1982

SDS

Webinar

SSH.COM

Exploring the Technology and Details of Securing FTP on z/OS and Beyond

SDS

About Us

sdsusa.com

Quality Mainframe Software since 1982

- Expert development & technical support teams based in Minneapolis, MN.
- 25+ products for z/OS, z/VM, z/VSE, and distributed platforms.
- Hundreds of organizations worldwide rely on SDS solutions.
- Focus on mainframe security and compliance.
- Cost savings and legacy tool replacements: DO MORE WITH LESS!
- Long-standing global partnerships complement SDS software.
- Recognized as cybersecurity trend-setter.

CYBERSECURITY 500 WORLD'S HOTTEST SECURITY COMPANIES

IBM Registered Business Partner

database TRENDS AND APPLICATIONS TREND-SETTING PRODUCTS for 2019

About SSH.com

Brief Company Introduction

Presenters

Jan Hlinovsky

Product Manager, Tectia SSH Server for z/OS

SSH.COM

A Data Set's Journey from z/OS to Unix and Back Again

Colin van der Ross

Sr. Systems Engineer

SDS

Demonstration of the z/OS SFTP topics covered by Jan

Tectia SSH Server for IBM z/OS

A Data Set's Journey from z/OS to Unix and Back Again

Jan Hlinovsky, Product Manager for Tectia on z
jan.hlinovsky@ssh.com

Storing text format data on a computer

- Computers operate with bits that are commonly collected into bytes. One byte, 8 bits, sometimes expressed as two hexadecimal values, can express 2^8=256 different values.
- In the 60s the ASCII code standardization and mainframe development were happening in parallel. ASCII code was 7 bit and its development stems from telegraph code. EBCDIC was 8 bit and is related to coding used with punch cards, and it was used in IBM system/360.

Storing text format data on a computer

- Computers operate with bits that are commonly collected into bytes. One byte, 8 bits, sometimes expressed as two hexadecimal values, can express 2^8=256 different values.
- In the 60s the ASCII code standardization and mainframe development were happening in parallel. ASCII code was 7 bit and its development stems from telegraph code. EBCDIC was 8 bit and is related to coding used with punch cards, and it was used in IBM system/360.

```
Binary: 0100 1110   0101 1010

Hex:        4 E         5 A

ASCII:       N           Z

EBCDIC:      +           !
```

Storing text format data on a computer

- Computers operate with bits that are commonly collected into bytes. One byte, 8 bits, sometimes expressed as two hexadecimal values, can express 2^8=256 different values.
- In the 60s the ASCII code standardization and mainframe development were happening in parallel. ASCII code was 7 bit and its development stems from telegraph code. EBCDIC was 8 bit and is related to coding used with punch cards, and it was used in IBM system/360.

```
Binary: 0100 1110   0101 1010

Hex:        4 E         5 A

ASCII:       N           Z

EBCDIC:      +           !
```

```
Binary: 1101 0101   1110 1001

Hex:        D 5         E 9

ASCII:   (invalid)   (invalid)

EBCDIC:      N           Z
```

SSH.COM

Common subsets and extensions

- There are several EBCDIC code pages, most common being 1047. There are also several 8 bit international character sets that have the ASCII code as a subset of the character set, sometimes called extended ASCII. For instance, ISO Latin 1, ISO8859-1.
- But the 256 possible values in one byte can only cover so much. Many languages have character sets that are much bigger -> need more bytes.
- These days, the Unicode standard covers most of the characters and symbols used in the world. There are several encoding methods, of which the UTF-8 is the most popular one. UTF8 is a variable length code (1-4 bytes per character) that also contains ASCII as a subset of the shortest (1 byte) characters.

- -> If you know what enconding your text uses, you can convert it to another encoding (that has the same characters), for instance with *iconv*

  Note: On z/OS, iconv works best with single byte code pages, e.g. to/from EBCDIC

What character set is in use?

- Typically, the character set in use is defined in the runtime environment, for instance in a Unix system in the locale and LC_* environment variables.
- Usually a system has a default character set defined, but users can decide to use a different one (and even change it at will).
- In other words, the character set in use in a text file is what was used when the file was saved.
- As a result, we can have an educated guess about the content type of a text file or data set, but we can not know it with 100% certainty. Therefore, when using the file later e.g. in file transfer, we want to give the user the ability to say how the content should be decoded.
- On z/OS, HFS files can have tags that help the operating system display text files correctly.

Newline conventions

- In addition to character sets, there is a higher level concept of ”newline”. Different systems use different encoding for this.
  - –z/OS data sets: record based, no concept of newline
  - –z/OS HFS files: EBCDIC newline character (NL, 0x25)
  - –Unix: ASCII linefeed character (\n, 0x0a)
  - –Windows: ASCII carriage return and linefeed combination (\r\n, 0x0d 0x0a)

```
$ cat testfile
line1
line2
$ hexdump –C testfile
00000000  6c 69 6e 65 31 0a 6c 69  6e 65 32 0a           |line1.line2.|
0000000c
$
```

What do file transfer programs do when transferring text files between different systems?

- Simple way: binary mode and "ascii" mode
  - binary mode: just transfer the data as is, and let the recipient handle any conversions if needed
  - ascii mode: perform the default conversion for newlines (and in some cases character sets)

- Works in most cases, but sometimes we need more control

- Example with Tectia: site parameters
  - I (TRANSFER_LINE_DELIMITER) - what delimiter is sent on the line
  - J (TRANSFER_FILE_LINE_DELIMITER) - what delimiter is used on the mainframe side
  - C (TRANSFER_CODESET) - what codeset is sent on the line
  - D (TRANSFER_FILE_CODESET) - what codeset is used on the mainframe side
    - The character sets supported with C and D are what **iconv** on the z/OS supports

Connecting with sftp and scp clients

- The scp client is a one line command to copy a file between two computers
- The sftp client tries to provide an FTP like user interface (even though the protocol is very different)
- Tectia command line tools include scpg3 (an scp client) and sftpg3 (sftp client)
  - support for direct dataset access
  - cryptography (cipher and HMAC) can be offloaded to CPACF or CEX
  - can be used to connect to any SSH server (for instance OpenSSH), translation is always performed on the mainframe side
  - can be used interactively or in JCL

z/OS to Unix

- For example, when using sftpg3 client to send a dataset from z/OS to a Unix machine:

```
sftp> lsite I=UNIX J=MVS
sftp> lsite C=ISO8859-1 D=IBM-1047
sftp> sput //DATASET file.txt
```

- What happens is that
  - On the z/OS side, the //DATASET is opened for reading, and there is an EBCDIC newline added (NL, 0x25) between the records in the buffer
  - The newline is converted into Unix style newline (0x0a)
  - The data in the records is converted from EBCDIC (IBM-1047) into ASCII/ISO-Latin1
  - On the Unix side, file.txt (in the current directory) is opened for writing
  - The data sent on the line and written to the file.txt is ISO8859-1 with Unix style newlines

Unix to z/OS

- Example: OpenSSH client on Linux sends a text file to an MVS dataset on z/OS that has Tectia Server

```
sftp> put file.txt /ftadv:I=unix,J=mvs,C=iso8859-1,D=ibm-1047/__DATASET
```

- The remote file string contains some additional information (file transfer advice string) that the server side processes
- The advice string tells the server that the incoming data has Unix style newlines and ISO-Latin1 character set in use, and conversion is requested.
- The destination is a dataset, __DATASET is an alternative format for //DATASET
- The newlines are converted to NL in the process, and finally the line format data is converted to record oriented format automatically because the destination is a data set. If the destination would be a file in the HFS filesystem, the newlines would be left as EBCDIC newline characters.

Special cases

- With z/OS to z/OS file transfers, sometimes we want to emulate FTP's RDW feature
  - need to use "F=record,J=MVS-FTP,X=bin"

- ASA printer control characters
  - need to use RECFM=FBA

- When the regular tables just won't do (i.e. if iconv does not have a translation)
  - option to use custom translation tables with TRANSFER_TRANSLATE_TABLE

Summary

- MVS datasets are record oriented, others (HFS, Unix, Windows) are stream oriented where newline in text is denoted by a special character or a combination of characters
- Character set conversion and newline conversion are two different things
- Character set conversion and newline conversion can change the size of a text file
- Easy mode: binary vs ascii
- Customizable options for newline conversion and character set conversion

SDS

SSH.COM

Colin van der Ross

Sr. Systems Engineer

SDS

Demonstration of the z/OS FTP topics covered by Jan

Agenda

- Commonly Used Advise Strings
- Additional file transfer information
- Demo of z/OS SFTP with SSH in action

SDS

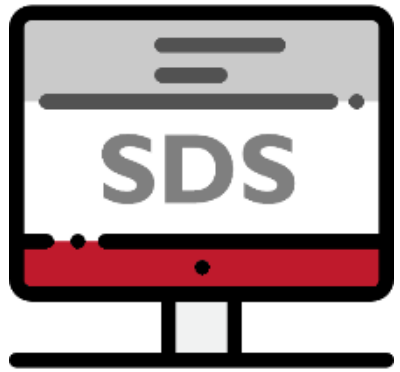
Have a Question?

SSH.COM

Would you like additional information?

info@sdsusa.com

(800) 443-6183
(763) 571-9000



www.sdsusa.com