ISPFSFTP V1.13

ISPFSFTP is a TSO utility to run sftp sessions in an interactive mode from foreground ISPF. It picks up the terminal emulation code page from ISPF and the user can enter the commands as if the session is set up with code page 01047. It supports to reach a target node via private and public key (preferred) or a password.

The tool consists of 10 files. ISPFSFTP and IBATSFTP should be placed into a REXX library in the SYSEXEC or SYSPROC library. The other eight files are placed into UNIX directory "/usr/local/bin" but you can change this. The file named "ispfsftp.unix" should be renamed to "ispfsftp" after copying. Routines "mvsalloc", "mvs2unix" and "unix2mvs" should be placed into a REXX library as well as they are also supported from TSO. Here is a suggestion how you can transfer and convert the files as needed.

- Transferred all files in binary to a USS (temporary) folder. So, they all arrive there in ASCII (ccsid 819) and need to be converted to 1047 next.
- Convert the two files to be placed to a PDS(E) under SYSPROC or SYSEXEC to 1047 within the USS directory...

```
$> iconv -f819 -t1047 ispfsftp > ispfsftp.1047
$> iconv -f819 -t1047 ibatsftp > ibatsftp.1047
$>
```

- Then go to the desired target PDS(E) and open empty member ISPFSFTP for editing and run command `copy /u/hering/workfolder/ispfsftp.1047` (as an example in my case). Then save the member and you are done. Do the same for IBATSFTP.
- Finally copy and convert the remaining files to "/usr/local/bin" (in authorized mode) or another location if desired.

```
$> iconv -f819 -t1047 ispfsftp.unix > /usr/local/bin/ispfsftp
$> iconv -f819 -t1047 ispfssh > /usr/local/bin/ispfssh
$> iconv -f819 -t1047 mvsalloc > /usr/local/bin/mvsalloc
$> iconv -f819 -t1047 mvs2unix > /usr/local/bin/mvs2unix
$> iconv -f819 -t1047 mvs2uss > /usr/local/bin/mvs2uss
$> iconv -f819 -t1047 ssh_askpass > /usr/local/bin/ssh_askpass
$> iconv -f819 -t1047 unix2mvs > /usr/local/bin/unix2mvs
$> iconv -f819 -t1047 uss2mvs > /usr/local/bin/uss2mvs
$>
```

If a password is needed it must be provided in an MVS PS data set named '*tsouser*.SFTP.PASSFILE'. The HLQ is the TSO userid of the user that opens the sftp session. This password file must be encoded in code page 01047. However, you can use string (at) instead of the @ character.

Important: Keep in mind, that the password strings must be provided according to ccsid 1047!

In the package Install-from-a-ZIP-file, that is provided here as well, you are guided to find your terminal emulation code page and the pipe (|) replacement character. Knowing both, you can see how to code passwords with special characters with following z/OS UNIX command.

```
echo passwd_with_sp_chars myprc iconv -f mytecp -t1047
```

Here myprc stays for your pipe replacement character and mytecp for your terminal emulation code page.

Following you see a sample contents for the PASSFILE:

```
hering@wtsc70oe.itso.ibm.com        :> password
hering(at)wtsc74oe                  :> password
heritst(at)wtsc70oe                 :> password
```

In this example ***password*** represents the password to be used.

You can provide changed/additional UNIX environment variables via allocated ddname SFTPENV or, if you do not use this ddname, in a file named '*tsouser*.SFTP.ENVVARS'. For example, you can change the standard PATH setting of ISPFSFTP. All these data sets should be created with record format VB. The password file must be defined as a VB data set.

In case of problems with passwords you can set an envvar ISPFSFTP_PWDTRACE=YES in the ENVVARS data set and create one further VB data set named '*tsouser*.SFTP.PWDTRACE'. This file contains a simple trace for your ISPFSFTP session just done and hopefully shows what causes the problem.

You can also change the standard password file via an IBATSFTP ddname or as an SFTPENV envvar as shown in the following.

- Example for specifying the PASSFILE via ddname ...

```
//PASSFILE DD DSNAME=HERING.SFTP.PASSFILE.VIADDN,DISP=SHR
```

- Example for specifying the PASSFILE via an environment variable ...

```
//SFTPENV  DD DATA,DLM=##
...
ISPFSFTP_PASSFILE=HERING.SFTP.PASSFILE.ENVVAR
...
##
```

Next the preference sequence for specifying the PASSFILE is listed. Most preferred is shown first.

- If a PASSFILE is specified via ISPFSFTP_PASSFILE envvar this is used. If no SFTPENV ddname is available file SFTP.ENVVARS under your HLQ is searched for that PASSFILE envvar setting.
- Next a PASSFILE ddname in an IBATSFTP job is used to determine the PASSFILE.
- Finally the standard name SFTP.PASSFILE under your HLQ is used if none of the previous choices is used.

An extra support has been added to allow specifying parts of FOTSxxxx messages to ignore these messages in decision whether a non-zero return code is set for the ISPFSFTP processing. You can use a data set named SFTP.IGNRMSGS under your HLQ or a ddname SFTPIGN in an IBATSFTP job. If you use a ddname in a job the data set is ignored if it exists.

Following you see a sample contents for the IGNRMSGS data set or ddname SFTPIGN.

```
FOTS9999 Xxxxxxxx Yyyyy
FOTS8888 Zzzzz
```

This means that if an FOTSxxxx message starts with "FOTS9999 Xxxxxxxx Yyyyy" or "FOTS8888 Zzzzz" then this message is ignored for setting a non-zero return code.

ISPFSFTP updates in V1.1

- Support has been added to geta job step RC<>0 in case of an error. This is based on SFTP messages shown. Output lines starting with OpenSSH FOTSnnnn force ISFTP019E.
- You can provide changed or additional UNIX environment variables via allocated ddname SFTPENV or, if not, in a file named 'TSOUSER.SFTP.ENVVARS'. This allows to put files like "ssh_askpass" at other places.
- Two new z/OS UNIX tools have been added to support easily placing a UNIX file to MVS. A simple routine named "mvsalloc" for allocation of an MVS data set has been created and a new tool "uss2mvs" for copying a UNIX file into an MVS data set.

ISPFSFTP updates in V1.2

- Changed normal stop processing to avoid using signal "sigkill". This previously forced getting a BPXP023I message written to the hardcopy log.

- Support has been added for using dd names in "uss2mvs" and "mvs2uss".

ISPFSFTP updates in V1.3

- You can specify now (at) instead of the 1047 @ character to minimize code page confusion.
- You can also specify option "-pu pw_user" to specify a fixed pass word userid in very special cases. This is normally never needed.
- A problem with changing the standard PATH setting of ispfsftp has been solved.

ISPFSFTP updates in V1.4

- By default ISPFSFTP is executed with option "StrictHostKeyChecking=yes" as using standard setting "ask" cannot work in this environment. To be able to add host keys automatically on an initial run you can force using value "no" via option "-nshkc" (No StrictHostKeyChecking).
- The batch interface routine IBATSFTP has been enhanced to allow continuation of lines in SFTPDATA in following lines. This is achieved by ending a line with a space delimited dash character ("-").

ISPFSFTP updates in V1.5

- A new routine named ispfssh has been added allowing to run ssh sessions from ISPF by calling ispfssh from separate ITSO utility named RXSHELL or within an RXBATCH or RXSBATCH based job.
- A second new routine named ispfsftp.unix has been added allowing to run sftp sessions from ISPF by calling it named ispfsftp again from separate ITSO utility RXSHELL or within an RXBATCH or RXSBATCH based job. This is a simpler alternative to ISPFSFTP and IBATSFTP, which are still preferable.

ISPFSFTP updates in V1.6

- An error in routine mvsalloc has been corrected. Also a new positional parameter has been introduced.
- A new REXX routine named unix2mvs has been added based on an previous version that had been created earlier already.
- A new REXX routine named mvs2unix has been added based on an previous version that had been created earlier already.

ISPFSFTP updates in V1.7

- An error in routine mvs2unix has been corrected.
- An error in routine unix2mvs has been corrected.

ISPFSFTP updates in V1.8

- A further positional parameter "mvs_disp" has been added to routine mvs2unix, and default is SHR. This avoids problems in case the data set is allocated shared already in parallel. (Many thanks to Hartmut Beckmann who suggested this change.)

ISPFSFTP updates in V1.9

- More details have been added into this small documentation reference.
- A further positional parameter "mvs_disp" has been added to routine unix2mvs.

ISPFSFTP updates in V1.10

- The default for positional parameter "mvs_disp" has been changed to SHR in routine unix2mvs in case of writing to a PO data set.

ISPFSFTP updates in V1.11

- The description about the PASSFILE has been improved. More information has been added how to install the package.
- A problem with utility unix2mvs on copying into a PDS(E) member has been corrected.

ISPFSFTP updates in V1.12

- New support for specifying an alternate password file has been added. You can use a ddname or an envvar in an IBATSFTP job.

ISPFSFTP updates in V1.13

- Fixed a small bug in routine "mvs2unix".
- New support for specifying FOTSxxxx messages to be ignored for setting a non-zero return code.

Syntax of command ISPFSFTP

Following the syntax of ISPFSFTP is shown:

```
ispfsftp [-nshkc] [-pu pw_user] [-u other_user] [sftp_options] userid@node
```

- You need to specify options if needed in the sequence as shown. You can add sftp options shown as "sftp_options" if desired.
- Option "-nshkc" means to run with setting "StrictHostKeyChecking=no". By default ISPFSFTP runs with setting "StrictHostKeyChecking=yes".
- Option "-pu" is normally never needed. Only use it if a password file is involved and you cannot easily pick up the right line. You can explicitly force to use a specific user entry in the password file.
- Using option "-u" allows to switch to another user before opening the sftp session. This described in more detail in the following.

Interactive and quasi-batch foreground sftp sessions

Following a sample sftp session is shown for reference:

```
ispfsftp hering@wtsc70oe.itso.ibm.com
 Connecting to wtsc70oe.itso.ibm.com...
 sftp>
cd /tmp
 sftp> cd /tmp
 sftp>
put .logout hering.test.copy
 sftp> put .logout hering.test.copy
 Uploading .logout to /SC70/tmp/hering.test.copy
 sftp>
quit
 sftp> quit
 ISFTP005I The sftp session ended normally.
 ***
```

If you have read authorization in class SURROGAT to profile BPX.SRV.HERITST you can switch to this other user and start the sftp session with that user:

```
ispfsftp -u heritst heritst@wtsc70oe.itso.ibm.com
```

There is a second routine named IBATSFTP that can be used when a set of sftp commands should be run in batch mode. You can either have a file allocated with ddname SFTPDATA or provide the name of an MVS data set (or member) containing the statements. The data set name must be provided fully qualified. The specification of the single quotes is optional. Here is an example:

```
ibatsftp 'hering.sftp.sessions(test003)'
 Connecting to wtsc70oe.itso.ibm.com...
 sftp> cd /tmp
 sftp> put .logout hering.test.copy
 Uploading .logout to /SC70/tmp/hering.test.copy
 sftp> quit
 ISFTP005I The sftp session ended normally.
 ***
```

If no data set is allocated with ddname SFTPDATA and you do not provide a name explicitly you simply get an error message as shown next.

```
ibatsftp
 IBATS004E DD name SFTPDATA and sftp commands file missing...
 ***
```

Using IBATSFTP in a batch job

Finally a batch example is shown that uses some more functions that get described afterwards. In this job an MVS data set is transferred to z/OS UNIX locally first and then to the remote node using sftp.

```
//IBATSFTP EXEC PGM=IKJEFT01,PARM=IBATSFTP
//SYSEXEC  DD DSNAME=HERING.REXX.EXEC,DISP=SHR
//SFTPDATA DD DATA,DLM=##
ispfsftp hering@wtsc70oe.itso.ibm.com
cd /tmp
! mvs2uss "'hering.test.file'" hering.test.file y y
! echo This is a looooooooooooooooooooooooooooooon -
g sftp sub-command using more than a line in the job.
put hering.test.file hering.test.file
!rm hering.test.file
quit
##
//SFTPENV  DD DATA,DLM=##
PATH=/u/hering/bin:/usr/local/bin:/bin
_EDC_ADD_ERRNO2=1
##
//CODEPAGE DD DATA,DLM=##
ISPFSFTP_CP=01141     < Terminal Emulation CP for the job and job log
##
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=VB
```

Following the job output data for ddname SYSTSPRT...

```
Connecting to wtsc70oe.itso.ibm.com...
sftp> cd /tmp
sftp> ! mvs2uss "'hering.test.file'" hering.test.file y y
sftp> ! echo This is a looooooooooooooooooooooooooooooooong sftp …
This is a looooooooooooooooooooooooooooooong sftp sub-command using …
sftp> put hering.test.file hering.test.file
Uploading hering.test.file to /SC70/tmp/hering.test.file
sftp> !rm hering.test.file
sftp> quit
ISFTP005I The sftp session ended normally.
READY
END
```

Additional utilities for copying files between MVS and z/OS UNIX

There are three small tools to be used for copy files from MVS to z/OS UNIX or vice versa. They can be called together with sftp as shown in the previous example with mvs2uss already. The usage of these small utilities is best chown by calling them from z/OS UNIX with no parameters as this prints sufficient help information.

Utility mvs2uss

```
$> mvs2uss
Syntax: /usr/local/bin/mvs2uss mvs_dsn uss_file sftp2win? cnv2ascii?

mvs_dsn   - MVS data set, for example: 'hering.test.data'
uss_file  - temporary USS file, for example: test.file
sftp2win  - sftp the uss_file to Windows? -- Y = Yes
cnv2ascii - convert data from 01141 to ascii? -- Y = Yes

$>
```

Notes:

- Routine "mvs2uss" should be placed to directory /usr/local/bin. It is for transferring an MVS data set to z/OS UNIX and placing it there in a format that just a binary transfer to the remote node needs to be done next.
- When transferring data to Windows (sftp2win set to **Y**) the line end chars are set to CRLF (x0D0A) instead of simply x0A.
- Conversion from EBCDIC 01141 to ASCII 00819 (ISO8859-1) is done when requested by setting cnv2ascii to **Y**. You can change the code page to your installation needs in the header part of the UNIX script file.

Utility mvsalloc

```
$> mvsalloc
Syntax: mvsalloc mvs_dsn sc,rf,rl,p,s<,0|nn>
mvs_dsn - MVS dsn; name without single quotes will be prefixed with the
userid.
     In case of UNIX enclose single quotes in a pair of double quotes.
sc - storclas, default STANDARD
rf - recfm, default VB
rl - lrecl, default 80 on RECFM=FB, 260 on RECFM=VB
p  - primary allocation in cylinders, default 1
s  - secondary allocation in cylinders, default 1
0  - create a PDSE PO data set
nn - create a PDS with a directory DIR(nn)
   - if not specified create a PS data set
$>
```

Notes:

- Routine "mvsalloc" should be placed to directory /usr/local/bin and a TSO REXX library as it is supported from z/OS UNIX and TSO. It is for easy allocation of a new MVS data set before copying a z/OS UNIX file to this MVS data set.
- You simply specify "rf" as VB or FB, for "rl" a record length can be specified. The data set is created with cylinders as the space unit. You can provide values for the primary ("p") and secondary ("s") allocation.
- Finally the number of directory blocks can be specified. If you specify "0" a PDSE is created, any other value forces the type to be PDS. If no number is provided a sequential data set is allocated.

Utility uss2mvs

```
$> uss2mvs
Syntax: /usr/local/bin/uss2mvs uss_file mvs_dsn sftpFwin? cnvFascii?
```

```
uss_file  - temporary USS file, for example: test.file
mvs_dsn   - MVS data set, for example: 'hering.test.data'
sftpFwin  - sftp the uss_file from Windows? -- Y = Yes
cnvFascii - convert data from ascii to 01141? -- Y = Yes

$>
```

Notes:

- Routine "uss2mvs" should be placed to directory /usr/local/bin. It is for transferring a z/OS UNIX file to an MVS data set or member.
- When transferring a file that came from Windows (sftpFwin set to **Y**) the line end characters are expected to be CRLF (x0D0A). Otherwise if file is in ASCII simply x0A is used.
- Conversion from ASCII 00819 (ISO8859-1) to EBCDIC 01141 is done when requested by setting cnv2ascii to Y. You can change the code page to your installation needs in the header part of the UNIX script file.

Utility unix2mvs

```
$> unix2mvs
Syntax: unix2mvs ussfile mvs_dsn <uss_le>,<uss_cp>,<mvs_cp>,<mvs_disp>
ussfile  - Source z/OS UNIX file
mvs_dsn  - Target z/OS MVS data set
           Name without single quotes will be prefixed with the userid.
           For UNIX enclose single quoted names in a pair of double quotes.
           Specify dd:ddname to address a ddname.
uss_le   - UNIX line end, typically crnl or nl, default nl
           (CR= x0D, ASCII NL= x0A, EBCDIC NL= x15)
uss_cp   - UNIX code page, default 00819 (=ISO8859-1)
mvs_cp   - MVS code page, default 01141 (=IBM-1141)
mvs_disp - MVS DISP for temp. allocations, default SHR for a PO, otherwise OLD
$>
```

Notes:

- Routine "unix2mvs" should be placed to directory /usr/local/bin and an TSO REXX library as it is supported from z/OS UNIX and TSO. It is for transferring a z/OS UNIX file to an MVS data set or member.
- It supports copying to a ddname (specified as dd:ddname) and also copying to a GDG version data set with relative specification like "'hering.gdg.testdata(0)'".
- Using MOD for "mvs_disp" is only valid for PS data sets and not for type PO. It allows to append data to an MVS sequential data set.

Utility mvs2unix

```
$> mvs2unix
Syntax: mvs2unix mvs_dsn ussfile <uss_le>,<uss_cp>,<mvs_cp>,<mvs_disp>
mvs_dsn  - Source z/OS MVS data set
           Name without single quotes will be prefixed with the userid.
           For UNIX enclose single quoted names in a pair of double quotes.
           Specify dd:ddname to address a ddname.
ussfile  - Target z/OS UNIX file
uss_le   - UNIX line end, typically crnl or nl, default nl
           (CR= x0D, ASCII NL= x0A, EBCDIC NL= x15)
uss_cp   - UNIX code page, default 00819 (=ISO8859-1)
```

```
mvs_cp   - MVS code page, default 01141 (=IBM-1141)
mvs_disp - MVS DISP for temp. allocations, default SHR
$>
```

Note:

- Routine "mvs2unix" should be placed to directory /usr/local/bin and to an TSO REXX library. It is for transferring a z/OS UNIX file to an MVS data set or member.
- It supports copying from a ddname (specified as dd:ddname) and also copying from a GDG version data set with relative specification like "'hering.gdg.testdata(0)'".

Using mvsalloc, unix2mvs and mvs2unix in an IBATSFTP job

Finally a batch example is shown that uses some more functions that get described afterwards. In this job an MVS data set is transferred to z/OS UNIX locally first and then to the remote node using sftp.

```
//IBATSFTP EXEC PGM=IKJEFT01,PARM=IBATSFTP
//SYSEXEC  DD DSNAME=HERING.REXX.EXEC,DISP=SHR
//SFTPDATA DD DATA,DLM=##
ispfsftp hering@wtsc70oe.itso.ibm.com
lpwd
! # Here a get command could place file test.ascii locally.
! mvsalloc "'hering.gdg.testdata(+1)'" standard,fb,80,1,1
! unix2mvs test.ascii "'hering.gdg.testdata(0)'" nl,819,1141
! mvs2unix "'hering.gdg.testdata(0)'" test.ascii.copy nl,819,1141
quit
##
//SFTPENV  DD DATA,DLM=##
PATH=/u/hering/bin:/usr/local/bin:/bin
_EDC_ADD_ERRNO2=1
##
//CODEPAGE DD DATA,DLM=##
ISPFSFTP_CP=01141     < Terminal Emulation CP for the job and job log
##
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*,LRECL=136,RECFM=VB
```

Following the job output data for ddname SYSTSPRT...

```
Connected to wtsc70oe.itso.ibm.com.
sftp> lpwd
Local working directory: /u/hering
sftp> ! # Here a get command could place file test.ascii locally.
sftp> ! mvsalloc "'hering.gdg.testdata(+1)'" standard,fb,80,1,1
sftp> ! unix2mvs test.ascii "'hering.gdg.testdata(0)'" nl,819,1141
sftp> ! mvs2unix "'hering.gdg.testdata(0)'" test.ascii.copy nl,819,1141
sftp> quit
ISFTP005I The sftp session ended normally.
READY
END
```

Syntax of command ispfssh

Following the syntax of ispfssh is shown:

```
ispfssh ssh_options_and_parameters
```

- Local code page support is provided by RXSHELL if started from ISPF or by RXSBATCH (even better than RXBATCH) if started from within a batch job. RXSHELL is available as an own ITSO utility.
- The password support of ISPFSFTP is automatically exploited.
- Note that you can provide an MVS PS data set named '*tsouser*.RXSHELL.ENVVARS' for RXSHELL to specify your desired settings for z/OS UNIX environment variables. The HLQ is your TSO userid.
- The additional new file to be used with name ispfsftp can be used similar and as alternative to the TSO command ISPFSFTP.

Following a sample ssh session from ISPF is shown for reference:

```
rxshell ispfssh hering@wtsc70oe
 FOTS2267 Pseudo-terminal will not be allocated because … not a terminal.
 >>
id
 uid=888(HERING) gid=2(SYS1) groups=1047(USSTEST)
 >>
pwd
 /u/hering
 >>
exit
 RXSHL009I The shell session ended normally.
 ***
```

Following a sample RXSBATCH JCL is shown for reference:

```
//RXBATCH  EXEC PGM=IKJEFT01,PARM=RXSBATCH
//SYSEXEC  DD DSNAME=HERING.UNIX.REXX.EXEC,DISP=SHR
//STDIN    DD DATA,DLM=##
ispfssh hilger@wtsc70oe
set -v
id; pwd
echo $PATH
rexx 'say Userid()'
##
//STDENV   DD DATA,DLM=##
PATH=/u/hering/bin:/usr/local/bin:/bin
##
//RXBPARM  DD DATA,DLM=##
 _RXBATCH_SWSU=0        < 0= no switch (default), 1= switch to SU mode
_RXBATCH_LOGIN=0        < 0= no login shell, 1= login shell (default)
  _RXBATCH_PGM=/bin/sh  < shell pgm "sh" or "tcsh", default is "/bin/sh"
_RXBATCH_CPIN=IBM-273   < code page of STDIN data, default is IBM-1047
_RXBATCH_CPOUT=CPIN     < STDOUT code page, default is STDIN CP (=CPIN)
##
//STDOUT   DD SYSOUT=*,LRECL=260,RECFM=VB
//SYSTSIN  DD DUMMY
//SYSTSPRT DD DUMMY    < Use "SYSOUT=*" instead in case of problems
```

Following the job output data for ddname SYSTSPRT...

```
FOTS2267 Pseudo-terminal will not be allocated because … not a terminal..
id; pwd
uid=1210(HILGER) gid=2(SYS1)
/u/hilger
echo $PATH
/u/hilger/bin:/usr/local/bin:/usr/lpp/Printsrv/bin:/bin
rexx 'say Userid()'
Rx> say Userid()
HILGER
```

Adding a new host to ~/.ssh/known_hosts

On using command "ispfsftp hering@wtsc70oe" you might get the following messages when wtsc70oe is not yet added to known_hosts because ispfsftp runs with "StrictHostKeyChecking=yes" by default.

```
FOTS1305 No RSA host key is known for wtsc70oe and you have requested
strict checking..
FOTS1370 Host key verification failed..
FOTS0841 Connection closed.
ISFTP006E Errors occurred; sftp gave RC= 255.
```

To overcome this and add the host to known_hosts you can use option "-nshkc" to run with setting "StrictHostKeyChecking=no" just for one time ("ispfsftp -nshkc hering@wtsc70oe"). You would get the following message if the host entry is added successfully.

```
FOTS2274 Warning: Permanently added 'wtsc70oe' (RSA) to the list of known
hosts.
```

You do not need to use this option again afterwards as the host is known now.

How to handle a possible hang-up situation

There is one situation where you could get into a hang-up situation. This is caused if you enter a z/OS UNIX command that results in a sub-command environment. ISPFSFTP cannot handle this based on its design. Therefore, a general UNIX shell environment is not allowed but it could be forced accidentally otherwise.

Following a sample with stopping the hang-up is shown for reference.

```
ispfsftp hering@wtsc70oe.itso.ibm.com
 Connecting to wtsc70oe.itso.ibm.com...
 sftp>
!
 ISFTP016W UNIX sub-environment is disallowed; use '! my_usscmd' instead.
 sftp>
! /bin/sh
 sftp> ! /bin/sh
[ esc ]
 !
 IRX0920I ENTER HI TO END, A NULL LINE TO CONTINUE, OR AN IMMEDIATE
COMMAND+ -
hi
 ISFTP017W Processing halted; exiting...
 ***
```

Entering "/bin/sh" forces to open a UNIX shell resulting in a hang-up of ISPFSFTP. You should press the escape

or PA1 key. Then you are prompted for an action. You should enter “hi” to halt processing and exit.

Notes:

- If this happens when using IBATSFTP you get out of the hang-up the same way.
- If you are running in a batch job simply cancel the job to get out of the hang-up.
- In general you should avoid getting in such a situation, of course.