Tectia Server 6.8 for IBM z/OS

Cookbook

08 April 2026

Tectia Server 6.8 for IBM z/OS : Cookbook

08 April 2026

SSH Communications Security Corporation
Karvaamokuja 2D, FI-00380 Helsinki, Finland

Table of Contents

1. About This Book 5
1.1. Who Should Use This Book 5
1.2. How to Use this Book 5
1.3. Related Documents 6
2. FTP-SFTP Conversion Through Socks Proxy 7
2.1. Files Used in This Example 7
2.2. Setting up FTP-SFTP Conversion 7
2.3. Running **ssh-socks-proxy** from JCL 9
3. Controlling File Transfers with File Transfer Advisor (FTADV) 11
3.1. FTADV in the File Transfer Command 11
3.2. File Transfer Profiles 12
4. Using Tectia Secure File Transfer Clients in Batch JCL 15
4.1. Tectia File Transfer Clients 15
4.2. File Transfer Examples 15
4.2.1. Putting an MVS data set to a remote Windows file 15
4.2.2. Fetching a remote file into an MVS data set 16
5. Managing JCL Jobs over SFTP with `filetype=JES` from Any Platform 19
5.1. Using File Transfer Advice String 19
5.1.1. Submitting a Job 19
5.1.2. Retrieving the Spool Output of a Job 20
5.1.3. Deleting a Job 20
5.1.4. Listing Jobs 20
5.2. Using File Transfer Profiles 21
6. Cryptographic Hardware Setup and Tuning 23
6.1. Configuring Ciphers and MACs 23
6.2. Access to Hardware Support to Generate Random Numbers 23
6.3. Enabling Cryptographic Hardware 24
6.4. Verifying that Cryptographic Hardware is Used 24
6.5. Optimizing Performance 25
7. ICSF PKCS11 Token Setup 27
7.1. z/OS ICSF PKCS11 Introduction 27

7.2. z/OS ICSF PKCS11 Token Dataset setup 28
7.2.1. Creating the TKDS 28
7.2.2. Define TKDS dataset to ICSF subsystem 29
7.3. Creating z/OS ICSF PKCS11 Token 29
7.3.1. Creating SAF Resource SO., USER. in class CRYPTOZ 29
7.3.2. Creating PKCS11 Token 29
7.4. Creating z/OS ICSF PKCS11 Object 30
7.4.1. Creating PKCS11 Object 30
7.5. Using z/OS ICSF PKCS11 Object 32
7.5.1. Use PKCS11 Objects in Tectia for z/OS programs 32
7.6. Deriving public key from z/OS ICSF PKCS11 Object 32
7.6.1. Derive public key from z/OS ICSF PKCS11 Object 32
A. Introduction to USS (UNIX) 35
A.1. UNIX File System 35
A.2. UNIX Files vs. MVS Data Sets 36
A.3. Referring to Data Sets 36
A.4. Setting Environment Variables in UNIX 37
A.5. Entering USS 37
A.6. File and Directory Permissions in UNIX 39
A.7. UID 40
A.8. MVS vs. UNIX Functional Comparison 41
A.9. Further Information 42
Index 43

Chapter 1 About This Book

1.1 Who Should Use This Book

This book is a collection of examples with instructions for performing different tasks with Tectia Server for IBM z/OS.

To take advantage of the examples in this book, you should already have Tectia Server for IBM z/OS installed and ready to be used. You can find instructions for doing that in the *Tectia Server for IBM z/OS Quick Start Guide*.

To fully utilize the examples presented in this book, you should be familiar with Unix System Services (USS) of z/OS and Unix concepts in general. If you are not previously familiar with USS (UNIX), we recommend you to read Appendix A before continuing to the examples.

1.2 How to Use this Book

The examples in this book build on each other using information or settings from previous sections. We recommend you to go through the examples in order, but you can skip around if you are confident enough in what you are doing.

You do not have to use the exact settings outlined in this guide, they are just examples.

The examples in this book are practical step-by-step instructions. If you want to find more detailed information about the topics covered in the examples, refer to the *Tectia Server for IBM z/OS Administrator Manual* and *User Manual*.

Note

Any information written in italics between '< >' (angle brackets) must be replaced with the information described between the angle brackets. For example, if your user ID was `smith`:

```
> cd <your_home_directory>
```


should be replaced with:

```
> cd /u/smith
```

1.3 Related Documents

For background information on the Tectia client/server solution, see the *Tectia Server for IBM z/OS Product Description*.

For quick installation instructions, see the *Tectia Server for IBM z/OS Quick Start Guide*.

For more detailed information and reference on the installation, configuration, and use of Tectia Server for IBM z/OS, see the *Tectia Server for IBM z/OS Administrator Manual*.

For detailed instructions on using the Tectia client tools on z/OS for secure system administration and secure file transfer, see the *Tectia Server for IBM z/OS User Manual*.

Chapter 2 FTP-SFTP Conversion Through Socks Proxy

2.1 Files Used in This Example

The following files will be modified or, if they do not already exist, created during this example:

`ssh-socks-proxy-config.xml`

The Tectia SOCKS Proxy configuration file `ssh-socks-proxy-config.xml` will in this example be created from the example configuration file `ssh-socks-proxy-config-example.xml`.

`<USERID>.FTP.DATA` (where `<USERID>` is your user ID) or `TCPIP.FTP.DATA`

If `<USERID>.FTP.DATA` does not exist, FTP will default to `TCPIP.FTP.DATA`. If you want these settings to be used only for `<USERID>`, you should create and use `<USERID>.FTP.DATA` as shown in this example.

`socks.conf`

This file can be called anything and placed anywhere (including as data set) as long as `<USERID>.FTP.DATA` or `TCPIP.FTP.DATA` point to the correct file. In this example `/opt/tectia/etc/socks.conf` will be created and used.

2.2 Setting up FTP-SFTP Conversion

In this example we assume that Tectia SOCKS Proxy is not yet running.

First, log on via a TN3270 emulator.

Take the following steps to set up FTP-SFTP conversion through Tectia SOCKS Proxy:

1. Create `ssh-socks-proxy-config.xml` if it does not already exist:

```
> cd /opt/tectia/etc
> cp ssh-socks-proxy-config-example.xml  ssh-socks-proxy-config.xml
```


2. Use **oedit** (or any other text editor of your choice) to edit the configuration file:

```
> oedit  ssh-socks-proxy-config.xml
```

Page down (by pressing **F8**) to the "Example filter rule" section to modify the SOCKS rule IP address to the desired IP address or addresses:

```
000079 <!-- Example filter rule used in FTP-SFTP conversion
000080      through SOCKS proxy. -->
000081 <!--
000082      <rule ip-address="10.1.2.3"
000083            ports="21"
000084            action="ftp-proxy"
000085            profile-id="id1"
000086            username-from-app="YES"
000087            hostname-from-app="YES"
000088            fallback-to-plain="NO" />
000089 -->
```

Remove the XML comments (`<!--` and `-->`) from around the `rule` element with `action="ftp-proxy"` and change `ip-address` to the IP address of incoming FTP requests or `".*"` for all incoming IP addresses:

```
000081
000082            <rule ip-address=".*"
000083            ports="21"
000084            action="ftp-proxy"
000085            profile-id="id1"
000086            username-from-app="YES"
000087            hostname-from-app="YES"
000088            fallback-to-plain="NO" />
000089
```

3. Create `socks.conf` (if it does not exist already) and modify it:

```
/opt/tectia/etc: > touch socks.conf
/opt/tectia/etc: > oedit socks.conf
```

Add these lines to the beginning of `socks.conf`:

```
sockd @=127.0.0.1 198.51.100 255.255.255.255
direct 0.0.0.0 0.0.0.0
```

In this example `198.51.100` is the FTP server IP address. Change this to match the IP address of your FTP server.

Note

Instead of `socks.conf`, it is also possible to store the SOCKS configuration in a data set.

4. You will need to edit `<USERID>.FTP.DATA` (if you only want the settings to be used for one user) or `TCPIP.FTP.DATA` (if you want the settings to be used for all users). In this example we create (if needed) and edit `<USERID>.FTP.DATA`.

   Go to ISPF from USS:

   (DO NOT press **enter** after issuing the following command)

   ```
   /opt/tectia/etc: > ISPF 3.4
   ```

   Press **F6** (make sure the F lock is on).

   If `<USERID>.FTP.DATA` does not exist, you need to create it using the characteristics of `TCPIP.FTP.DATA`.

   Add the following lines to `<USERID>.FTP.DATA` if they do not already exist:

   ```
   SOCKSCONFIGFILE /opt/tectia/etc/socks.conf
   FWFRIENDLY TRUE
   ```

5. Now all unsecured FTP traffic to IP address `198.51.100` on port `21` will be secured with SSH SFTP:

   ```
   /opt/tectia/etc ftp 198.51.100
   Using '<USERID>.FTP.DATA' for local site configuration parameters.
   IBM FTP CS V1R12
   Connecting to:  198.51.100 port: 21.
   220-------------------------------------
   220--- SSH Tectia FTP-SFTP Conversion ---
   220-------------------------------------
   220 Your FTP connection will be SECURED!
   ```

   If you do not see the "`SSH Tectia FTP-SFTP Conversion`" message your connection is not secure.

2.3 Running ssh-socks-proxy from JCL

You can use this JCL (after modifying it to suit your requirements) to run the SOCKS Proxy:

```
//USERSSP  JOB ,,CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),
//             NOTIFY=&SYSUID
//*
//* Tectia SOCKS Proxy using catalogued proc
//*
//SOXPROX EXEC PROC=SSHSP
//* Override sshenv, for example
//STDENV   DD  DSN=<HLQ>.V680.PARMLIB(SSHENV),DISP=SHR
//
```


Chapter 3 Controlling File Transfers with File Transfer Advisor (FTADV)

With file transfer advisor (FTADV) any SFTP client, on any platform, can tell the Tectia z/OS SFTP server exactly how to process and handle the file transfer. A lot like SITE commands, FTADV works the same no matter what platform the client runs on.

In the following sections we describe two different ways to take advantage of FTADV for you convenience: entering the FTADV data directly into the file transfer command, or using file transfer profiles in the FTADV configuration file.

3.1 FTADV in the File Transfer Command

Enter the FTADV data directly into the file transfer command in a *file transfer advice string*.

For example, to make sure that the data in the file that is being fetched is treated as binary, use the file transfer advice string `/FTADV:X=BIN/`:

```
sftp> get /FTADV:X=BIN/test.txt
```

Note

This overrides the settings defined in the `ssh_ftadv_config` file described in Section 3.2.

The following table lists some of the commonly used file transfer advice string names with their abbreviations, possible values and short descriptions. Consult the *Tectia Server for IBM z/OS User Manual* for a complete list of the available file transfer advice string names and their detailed descriptions.

Table 3.1. Commonly Used Advice Strings

Advice String	Description
`BLKSIZE|B|BLOCKSI=`*`size`*	Specifies the maximum block size.
`CONDDISP|CO =CATLG|UNCATLG|KEEP|DELETE`	Specifies the disposition of the output file when a file transfer ends prematurely.
`FILE_STATUS|STATUS=NEW|MOD|SHR|OLD`	Defines the status of a data set. If entered, the value will be used when allocating the data set.
`FILETYPE|FILET=SEQ|JES`	Specifies whether to interface with the file system or with the z/OS Job Entry Subsystem (JES).
`FIXRECFM|FI=`*`length`*	The data set organization is set to FB and the fixed record length is set to *`length`*.
`LIKE=`*`like`*	Specifies the name of a model data set from which the `RECFM`, `BLKSIZE`, and `LRECL` attributes are to be copied.
`LRECL|R|LR=`*`length`*	Maximum record length or fixed record length
`NORMDISP|NOR =CATLG|UNCATLG|KEEP|DELETE`	Specifies the data set disposition to be used after a file transfer that ends normally.
`PRIMARY_SPACE|PRI=`*`space`*	Primary space allocation for a data set.
`PROFILE|P|PROF=profile`	The file transfer profile specifies the named profile used for the file transfer.
`RECFM|O|REC=`*`recfm`*	Specifies the data set organization.
`SPACE_UNIT|SU =BLKS|TRKS|CYLS|AVGRECLEN`	Specifies the unit of space allocation for a data set.
`TRANSFER_CODESET|C|CODESET=`*`codeset`*	Specifies the code set of the data during transfer.
`TRANSFER_FILE_CODESET|D|FCODESET =`*`codeset`*	Specifies the code set of the data in the data set.
`TRANSFER_FILE_LINE_DELIMITER |J|FLDELIM =UNIX|MVS|MVS-FTP|DOS|MAC|NEL`	Specifies the newline convention used in the (source or destination) file.
`TRANSFER_FORMAT|F|FORMAT =LINE|STREAM|RECORD`	Specifies the transfer format.
`TRANSFER_LINE_DELIMITER|I|LDELIM =UNIX|MVS|MVS-FTP|DOS|MAC|NEL`	Specifies the newline convention used in the data that is transferred over the connection.
`TRANSFER_MODE|X|MODE=BIN|TEXT`	Specifies whether code set and line delimiter conversions are performed.

3.2 File Transfer Profiles

Edit the global or local user `ssh_ftadv_config` file to include a named or filename-matched profile.

We provide a sample configuration file (`/opt/tectia/etc/ssh_ftadv_config.example`). It is a good idea to copy this file to the directory you want and edit it to match your preferences.

- `/opt/tectia/etc/ssh_ftadv_config` – Global FTADV configuration file. This file is in effect for all users unless the user in question has a local FTADV configuration file.
- `$HOME/.ssh2/ssh_ftadv_config` – User local FTADV configuration file. If present, this file overrides the Global FTADV configuration file.

In the FTADV configuration file of your choice you can use one of the following types of profiles:

- **Named profile:** Only used when specified via `/FTADV:P=<profilename>/`.

  The following example profile converts text files from Unix to MVS. ASCII is converted to EBCDIC.

  ```
  %UNIX          X=text,
                 F=line,
                 C=iso8859-1,
                 D=ibm-1047
  ```

  Individual attributes can be overridden like this: `/FTADV:P=UNIX,C=ibm-1047/`. (The transfer code set attribute `C` is overridden).

- **Filename-matched profile:** Any file that matches the regular expression will use the specified settings. The first match is used.

  The following example profile matches files that have one of the extensions listed in parentheses. Code set conversion from ASCII to EBCDIC is performed.

  ```
  # Match text files that end with `.' and extension.
  .*\\.(txt|TXT|c|C|h|H|log|LOG|conf|CONF)$
      X=text,
      F=line,
      C=iso8859-1,
      D=ibm-1047
  ```

Take a look at Section 5.2 for a good practical of matched profiles.

The following table describes the named transfer profiles that are defined in `/opt/tectia/etc/ssh_ftadv_config.example`.

Table 3.2. Named File Transfer Profiles

Profile Name	Included FTADV Attributes	Usage
`UNIX`	`X=text, F=line, C=iso8859-1, D=ibm-1047`	Text files from Unix to z/OS
`WIN`	`X=text, F=line, C=iso8859-1, D=ibm-1047, I=dos, J=mvs`	Text files from Windows to z/OS
`ZOS`	`X=text, F=line`	Text files from z/OS to another z/OS
`FB80`	`X=text, F=line, C=iso8859-1, D=ibm-1047, O=fb, R=80`	Text files between Unix and z/OS. Data sets created in z/OS will have fixed blocked format with 80 byte records.
`REC`	`F=record, R=1024`	Preserves record length information. Data sets created in z/OS will have variable blocked format with a maximum record length of 1024 bytes.
`BIN`	`X=bin, F=stream`	Binary file transfers
`%`	`X=bin, F=stream`	Disables data set pattern matching

Chapter 4 Using Tectia Secure File Transfer Clients in Batch JCL

4.1 Tectia File Transfer Clients

Tectia client tools for z/OS contain two file transfer applications, **scpg3** and **sftpg3**:

- **scpg3** is a secure replacement for remote copy (rcp) and provides easy secure non-interactive file transfers.
- **sftpg3** is a secure replacement for FTP and provides a user interface for interactive file transfers and a batch mode for unattended file transfers.

scpg3 and **sftpg3** are executed in JCL by BPXBATCH.

Note that when running the client programs in JCL, the `_BPX_SHAREAS` environment variable must be set to `no`.

4.2 File Transfer Examples

We provide sample JCL for different types of secure file transfers in *<HLQ>*`.V680.SAMPLIB`. Two basic unattended file transfers using **scpg3** and **sftpg3** are presented here.

4.2.1 Putting an MVS data set to a remote Windows file

In this example (`SCPPUT1` from `SAMPLIB`), **scpg3** is executed to copy a data set to a remote file (*`test.list`*), converting the code set from IBM-1047 to ISO8859-1 and records to CR-LF delimited lines.

The stdout and stderr message files are printed to `SYSOUT`. Required environment variables are supplied in `SSHENV` via `STDENV DD`. Modify the DD statement according to your requirements.

```
//SCPPUT1 EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT
//STDPARM  DD  *
PGM /opt/tectia/bin/scpg3
```



```
  /ftadv:C=ISO8859-1,D=IBM-1047,I=DOS,J=MVS//__HLQ.TEST.LIST
  user@remote:test.list
//STDENV   DD  DSN=<HLQ>.V680.PARMLIB(SSHENV),DISP=SHR
//STDOUT   DD  SYSOUT=*
//STDERR   DD  SYSOUT=*
//STDIN    DD  DUMMY
//
```

The same file transfer can be carried out using **sftpg3** in batch mode (`-B` option):

```
//SFTPPUT1 EXEC PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT
//STDPARM  DD  *
PGM /opt/tectia/bin/sftpg3 -B //DD:STDIN
  user@remote
//STDENV   DD  DSN=<HLQ>.V680.PARMLIB(SSHENV),DISP=SHR
//STDOUT   DD  SYSOUT=*
//STDERR   DD  SYSOUT=*
//STDIN    DD  *
sput /ftadv:C=ISO8859-1,D=IBM-1047,I=DOS,J=MVS//__HLQ.TEST.LIST test.list
//
```

4.2.2 Fetching a remote file into an MVS data set

This example (`SCPGET` from `SAMPLIB`) executes **scpg3** and copies a remote file (*`file.bin`*) into a data set (*`//'USER.TEST.BINFILE'`*). If the data set does not exist, it is created with default values `recfm VB` and `lrecl 1024`.

The stdout and stderr message files are printed to `SYSOUT`. Required environment variables are supplied in `SSHENV` via `STDENV DD`. Modify the DD statement according to your requirements.

```
//SCPGET  EXEC PGM=BPXBATSL,REGION=0M
//STDPARM  DD  *
PGM /opt/tectia/bin/scpg3
  user@remote:file.bin
  //'USER.TEST.BINFILE'
//STDENV   DD  DSN=<HLQ>.V680.PARMLIB(SSHENV),DISP=SHR
//STDOUT   DD  SYSOUT=*
//STDERR   DD  SYSOUT=*
//STDIN    DD  DUMMY
//
```

The same file transfer can be carried out using **sftpg3** in batch mode (`-B` option):

```
//SFTPGET  EXEC PGM=BPXBATSL,REGION=0M
//STDPARM  DD  *
PGM /opt/tectia/bin/sftpg3 -B //DD:STDIN
  user@remote
//STDENV   DD  DSN=<HLQ>.V680.PARMLIB(SSHENV),DISP=SHR
//STDOUT   DD  SYSOUT=*
//STDERR   DD  SYSOUT=*
//STDIN    DD  *
```



```
sget file.bin //'USER.TEST.BINFILE'
//
```


Chapter 5 Managing JCL Jobs over SFTP with `filetype=JES` from Any Platform

Tectia Server for IBM z/OS provides the functionality for managing JCL jobs remotely. The JCL scripts are transferred to the z/OS MVS Job Entry Subsystem (JES) using SFTP. The JES interface of Tectia Server for IBM z/OS supports submitting and deleting jobs, as well as receiving the spool output and displaying the status of jobs.

File transfer advisor (FTADV) `filetype=JES` is required to interface with JES instead of the file system. For more information on FTADV, see Chapter 3.

In the following sections we show you how to interface with JES using file transfer advice strings and file transfer profiles.

5.1 Using File Transfer Advice String

5.1.1 Submitting a Job

In this example it is assumed that the JCL script `br14.jcl` with the following contents is stored in the directory `/home/user1/src/jcl/` on a Unix host:

```
//USERJ0  JOB ,,CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),
//            NOTIFY=&SYSUID
//*
//STEP00  EXEC PGM=IEFBR14
//
```

Submit a job for execution using **sftpg3** and receive a notification of the ID assigned to the submitted job:

```
$ sftpg3 user1@mf_server ❶
sftp> sput /home/user1/src/jcl/br14.jcl /ftadv:filetype=jes,c=ISO8859-1,d=IBM-1047/ ❷
br14.jcl                            |   100B |      29B/s | TOC: 00:00:03 | 100%
07.57.19 JOB03198  $HASP100 USERJ0   ON INTRDR              FROM STC03197 USER17
```



```
07.57.20 JOB03198  IRR010I  USERID USER1   IS ASSIGNED TO THIS JOB.
JOBID=JOB03198 ❸
```

❶ Open an SFTP session from your client to the target server (in this example `mf_server`).

❷ Submit the job (`/home/user1/src/jcl/br14.jcl`) using **sput**. Use a file transfer advice string to set file type to JES and to specify code set conversion. In this example the code set is ISO8859-1 during the transfer and the server should store the data set with the IBM-1047 code set.

❸ At the end of the output you can see the job ID (`JOB03198`) that was assigned to the job.

5.1.2 Retrieving the Spool Output of a Job

To retrieve the spool output of a submitted job, run the **get** command with the job's ID, and specify the file type and code set conversion using a file transfer advice string:

```
sftp> get /ftadv:filetype=JES,C=ISO8859-1,D=IBM-1047/JOB03198
JOB03198                          |  1.1kB |     431B/s | TOC: 00:00:02 | 100%
```

5.1.3 Deleting a Job

To delete a job (in this example `JOB03198`), use the **rm** command and an advice string to set file type to JES:

```
sftp> rm /ftadv:filetype=JES/JOB03198
```

5.1.4 Listing Jobs

To display the status of all the jobs that are on the JES spool for your user ID, enter the following command:

```
sftp> ls /ftadv:filetype=JES/
```

To list jobs in the long name format, enter:

```
sftp> ls -l /ftadv:filetype=JES/
```

To list the contents of a specific job (in this example `JOB03419`) in the long name format:

```
sftp> ls -l /ftadv:filetype=JES/JOB03419/ ❶
/FTADV:filetype=JES//u/home/user1/JOB03419/:
Volume Referred    Recfm Lrecl BlkSz Dsorg    Space  Dsname
JOB03419 USER1   USERJ0   A        J      0000
  0002          JES2     JESMSGLG    18       1048 UA       133
  0003          JES2     JESJCL       6        299 V        136
  0004          JES2     JESYSMSG     9        559 VA       137
```

❶ Note the required trailing slash after the job ID.

5.2 Using File Transfer Profiles

1. Copy the example file transfer profile file `/opt/tectia/etc/ssh_ftadv_config.example` to `ssh_ftadv_config`. (You can skip this step if `/opt/tectia/etc/ssh_ftadv_config` already exists.)

   ```
   > cd /opt/tectia/etc
   > cp ssh_ftadv_config.example ssh_ftadv_config
   ```

2. Use **oedit** or any other text editor of your choice to edit the `ssh_ftadv_config` file:

   ```
   > oedit ssh_ftadv_config
   ```

3. Add the following lines before the line "`# Match all other files.`":

   ```
   # Match files that end with '.jcl'
   .*\\.(jcl)$
         FILETYPE=JES,
         X=text,
         F=line,
         C=iso8859-1,
         D=ibm-1047

   # Match files that start with JOB
   .\\JOB*
         FILETYPE=JES,
         X=text,
         F=line,
         C=iso8859-1,
         D=ibm-1047
   ```

4. Press **F3** to save and close the file.

Now you can securely submit jobs to JES from any platform by simply doing a **put** of any file that ends in `.jcl` and get the results for that job by doing a **get** with a job ID.

1. Create some sample JCL files in proper JCL format on the machine where your SSH client resides. This can be any platform including z/OS, Windows and different UNIX platforms.

   You can use the following JCL for testing purposes:

   ```
   //USERJ0 JOB ,,CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),
   // NOTIFY=&SYSUID
   //*
   //STEP00 EXEC PGM=IEFBR14
   //
   ```

   For this example we named the file `br14.jcl`.

2. Connect to the z/OS SFTP server where you edited the `ssh_ftadv_config` file:


```
> sftpg3 <my-zos-host>
```

Make sure that the server is running and you are using the correct port.

3. Enter the following command from the **sftp** command prompt:

```
sftp> put br14.jcl
```

At the end of the output you can see the job ID. In this example, the job ID is `JOB09291`.

```
br14.jcl
| 116B | 110B/s | TOC: 00:00:01 | 100%
02.31.34 JOB09291 $HASP100 USERJ0 ON INTRDR
FROM STC09290 MACH8
02.31.34 JOB09291 IRR010I USERID USER IS ASSIGNED TO THIS JOB.
JOBID=JOB09291
```

4. To get the results for `JOB09291`, enter the following command from the **sftp** command prompt:

```
sftp> get JOB09291
```

The job results will be stored in the current **sftp** local directory in a file named `JOB09291`.

Chapter 6 Cryptographic Hardware Setup and Tuning

6.1 Configuring Ciphers and MACs

For best performance, prune the cipher and MAC algorithms in the server configuration file to only those that are supported by the cryptographic hardware. If a client suggests an algorithm that is not supported by the cryptographic hardware, software cryptography will be used.

Example: Ciphers and MACs in the server configuration file `sshd2_config`

```
Ciphers           aes128-cbc,aes192-cbc,aes256-cbc,3des-cbc
MACs              hmac-sha1,hmac-sha1-96
```

The example `sshd2_config` configuration file lists the algorithms that are used by default. For a list of all the supported algorithms, see the *Administrator Manual*.

6.2 Access to Hardware Support to Generate Random Numbers

Tectia for z/OS client and server programs use ICSF Random Number Generation service to collect random numbers.

Make sure users have access to the `ICSF CSFRNG` (Integrated Cryptographic Service Facility random number generate) service:

```
RDEFINE CSFSERV CSFRNG UACC(NONE)
PERMIT CSFRNG CLASS(CSFSERV) ID(*) ACCESS(READ)
SETROPTS RACLIST(CSFSERV) REFRESH
```


6.3 Enabling Cryptographic Hardware

To enable cryptographic hardware you need to enable the following `CSFSERV` profiles for all client and server IDs in RACF:

```
RDEFINE CSFIQA CLASS(CSFSERV) UACC(NONE)
RDEFINE CSF1TRC CLASS(CSFSERV) UACC(NONE)
RDEFINE CSF1TRD CLASS(CSFSERV) UACC(NONE)
RDEFINE CSF1SKE CLASS(CSFSERV) UACC(NONE)
RDEFINE CSF1SKD CLASS(CSFSERV) UACC(NONE)
RDEFINE CSFOWH CLASS(CSFSERV) UACC(NONE)

PERMIT CSFIQA CLASS(CSFSERV) ID(*) ACCESS(READ)
PERMIT CSF1TRC CLASS(CSFSERV) ID(*) ACCESS(READ)
PERMIT CSF1TRD CLASS(CSFSERV) ID(*) ACCESS(READ)
PERMIT CSF1SKE CLASS(CSFSERV) ID(*) ACCESS(READ)
PERMIT CSF1SKD CLASS(CSFSERV) ID(*) ACCESS(READ)
PERMIT CSFOWH CLASS(CSFSERV) ID(*) ACCESS(READ)
SETROPTS CLASSACT(CSFSERV)
SETROPTS RACLIST(CSFSERV) REFRESH
```

If possible, avoid defining the following SAF/RACF profile. Otherwise you must grant READ access to this profile for all client and server IDs:

```
CLASS(CRYPTOZ) CLEARKEY.SYSTOK-SESSION-ONLY
```

6.4 Verifying that Cryptographic Hardware is Used

To verify that cryptographic hardware is being used, set the debug level for `SecShPlugin*ZosIcsf` to 4. Setting all debug to level 4 would have the same result, but you would end up with a large amount of data to look through.

You can use this command from USS to verify that cryptographic hardware is enabled:

```
> sshg3 -DSecShPlugin*ZosIcsf=4 127.0.0.1
```

The command should produce the following type of output without CEX:

```
Setting debug level string to 'SecShPlugin*ZosIcsf=4'.
...

ssh_secsh_plugin_init: Card IO Threshold = 65536
state_determine: Hardware for 3des-cbc: ICSF-CPACF
state_determine: Hardware for aes128-cbc: ICSF-CPACF
state_determine: Hardware for aes192-cbc: ICSF-CPACF
state_determine: Hardware for aes256-cbc: ICSF-CPACF
state_determine: Hardware for aes128-ctr: ICSF-CPACF
state_determine: Hardware for aes192-ctr: ICSF-CPACF
state_determine: Hardware for aes256-ctr: ICSF-CPACF
```



```
state_determine: Hardware for aes128-ecb: ICSF-CPACF

ssh_secsh_plugin_init: Card HMAC generate = FALSE
state_determine: Hardware for hmac-sha1: ICSF-CPACF
state_determine: Hardware for hmac-sha1-96: ICSF-CPACF
state_determine: Hardware for hmac-sha256@ssh.com: ICSF-CPACF
state_determine: Hardware for hmac-sha2-256: ICSF-CPACF
state_determine: Hardware for hmac-sha256-2@ssh.com: ICSF-CPACF
state_determine: Hardware for hmac-sha224@ssh.com: ICSF-CPACF
state_determine: Hardware for hmac-sha384@ssh.com: ICSF-CPACF
state_determine: Hardware for hmac-sha2-512: ICSF-CPACF
state_determine: Hardware for hmac-sha512@ssh.com: ICSF-CPACF
```

The command should produce the following type of output when CEX is enabled:

```
Setting debug level string to 'SecShPlugin*ZosIcsf=4'.

...

ssh_secsh_plugin_init: Card IO Threshold = 0
state_determine: Hardware for 3des-cbc:   ICSF-COP
state_determine: Hardware for aes128-cbc: ICSF-COP
state_determine: Hardware for aes192-cbc: ICSF-COP
state_determine: Hardware for aes256-cbc: ICSF-COP
state_determine: Hardware for aes128-ctr: ICSF-COP
state_determine: Hardware for aes192-ctr: ICSF-COP
state_determine: Hardware for aes256-ctr: ICSF-COP
state_determine: Hardware for aes128-ecb: ICSF-COP

ssh_secsh_plugin_init: Card HMAC generate = TRUE
state_determine: Hardware for hmac-sha1: ICSF-COP
state_determine: Hardware for hmac-sha1-96: ICSF-COP
state_determine: Hardware for hmac-sha256@ssh.com: ICSF-COP
state_determine: Hardware for hmac-sha2-256: ICSF-COP
state_determine: Hardware for hmac-sha256-2@ssh.com: ICSF-COP
state_determine: Hardware for hmac-sha224@ssh.com: ICSF-COP
state_determine: Hardware for hmac-sha384@ssh.com: ICSF-COP
state_determine: Hardware for hmac-sha2-512: ICSF-COP
state_determine: Hardware for hmac-sha512@ssh.com: ICSF-COP
```

6.5 Optimizing Performance

To optimize performance you can define the `CSFOWH` (hash) and `CSFRNG` (random number) profiles in the `XFACILIT` class. This will disable SAF/RACF checks for these profiles. ICSF uses CPACF instructions for these anyway and CPACF instructions cannot be protected by SAF/RACF. Consult your security team to make sure this is acceptable.

Example: RACF instructions


```
RDEFINE CSF.CSFSERV.AUTH.CSFOWH.DISABLE CLASS(XFACILIT) UACC(READ)
RDEFINE CSF.CSFSERV.AUTH.CSFRNG.DISABLE CLASS(XFACILIT) UACC(READ)
SETROPTS CLASSACT(XFACILIT)
SETROPTS RACLIST(XFACILIT) REFRESH
```


Chapter 7 ICSF PKCS11 Token Setup

7.1 z/OS ICSF PKCS11 Introduction

z/OS ICSF PKCS11 token is a virtual key store for each z/OS user. A PKCS11 token can store 3DES, AES, RSA, ECDSA, ED25519 keys for key generation, cipher, hash, sign and verify operation. ICSF PKCS11 token can be referred by token name, e.g. SSZ.SSHD2.HOSTKEY, SSZ.TSTUSR.SSH2.

An object is key value stored in a PKCS11 token. An object can be referred by token_name/object_id in Tectia for z/OS programs, e.g. SSZ.SSHD2.HOSTKEY/00000001, SSZ.TSTUSR.SSH2/000000AB.

z/OS PKCS11 tokens are stored in a z/OS token data set and configured to ICSF started task using TKDSN option.

The access of a z/OS PKCS11 token is protected by SAF and audited by ICSF started task. SAF profile prefixed with USER and SO in class CRYPTOZ are used to protect the access of a z/OS PKCS11 token, e,g SO.SSZ.SSHD2.HOSTKEY, USER.SSZ.SSHD2.HOSTKEY. SAF profile prefixed with SO acts as a security officer to manage a z/OS PKCS11 token. SAF profile prefixed with USER acts as an user to use object key in z/OS PKCS11. ICSF option AUDITAUDITPKCS11USG can be configured to control audit of PKCS11 tokens/objects.

PKCS11 token/object can be managed via ICSF Token Management panel. The ICSF Token Management panel can be accessed from ICSF main menu option 5 (ICSF Utilities) and then in ICSF Utilities menu option 7 (PKCS11 TOKEN).

Note

Tectia for z/OS programs use mainframe message security assist extension 9 (MSAE9) instructions to perform ECDSA and Ed25519 elliptic curve functions. Mainframe model z15 and onward contains this feature. If the MSAE9 feature is not available, Tectia for z/OS programs will only support RSA algorithm in z/OS PKCS11 token.

```
CSFTBR00 ------------- ICSF Token Management - Main Menu -----------
OPTION ===>
```



```
Enter the number of the desired option.

  1  Create a new token
  2  Delete an existing token
  3  Manage an existing token
  4  List existing tokens

  Full or partial token name: _____________________________
```

7.2 z/OS ICSF PKCS11 Token Dataset setup

ICSF PKCS11 Token Dataset is a VSAM data set to store PKCS11 tokens and objects. If token data set (TKDS) is enabled in ICSF subsystem, TKDS dataset name is shown via operator command 'D ICSF,KDS' and option 'PKCS11 TOKEN - Manage PKCS11 tokens in the TKDS' is shown in ISPF 'ICSF - Utilities' panel.

```
D ICSF,KDS
CSFM668I 12.44.03 ICSF KDS
  CKDS  CSF.SCSFCKDS
    FORMAT=FIXED                SYSPLEX=N  MKVPs=DES AES
    DES MKVP date=2025-11-18 04:39:07
    AES MKVP date=2025-11-18 04:39:07
  PKDS  CSF.SCSFPKDS
    FORMAT=KDSR                 SYSPLEX=N  MKVPs=RSA ECC
    RSA MKVP date=2025-11-18 04:39:08
    ECC MKVP date=2025-11-18 04:39:08
  TKDS  CSF.SCSFTKDS
    FORMAT=VARIABLE             SYSPLEX=N  MKVPs=None
```

```
CSFUTL00 ---------------------- ICSF - Utilities -----------
OPTION ===>

Enter the number of the desired option.

  1  ENCODE       -  Encode data
  2  DECODE       -  Decode data
  3  RANDOM       -  Generate a random number
  4  CHECKSUM     -  Generate a checksum and verification patterns
  5  CKDS KEYS    -  Manage keys in the CKDS
  6  PKDS KEYS    -  Manage keys in the PKDS
  7  PKCS11 TOKEN -  Manage PKCS11 tokens in the TKDS
```

If TKDS is not enabled in ICSF subsystem, following steps must be performed.

7.2.1 Creating the TKDS

User can refer to section 'Creating the TKDS' of Chapter 2 'Installation, initialization, and customization' in z/OS: z/OS ICSF System Programmer's Guide manual to create TKDS dataset.

7.2.2 Define TKDS dataset to ICSF subsystem

The TKDS data set name must be specified on TKDSN parameter of the options data set when you start the ICSF subsystem.

7.3 Creating z/OS ICSF PKCS11 Token

Each PKCS11 token is protected by SAF CRYPTOZ class. There are two resources defined for each token. The resource USER.token-name controls the access of the User role to the token. The resource SO.token-name controls the access of the SO role to the token. SO user is responsible to create/delete a PKCS11 token when USER user is the user to use objects in a PKCS11 token.

For dmonstration purpose, a z/OS normal user with TSO and OMVS (non zero UID) segment is created and called SSZKMGR and acts as SO role to z/OS ICSF PKCS11 token used for Tectia for z/OS programs.

7.3.1 Creating SAF Resource SO., USER. in class CRYPTOZ

Create SAF resource for token SSZ.SSHD2.HOSTKEY for storing private key objects for sshd2 hostkey.

```
RDEFINE CRYPTOZ SO.SSZ.SSHD2.HOSTKEY UACC(NONE)
RDEFINE CRYPTOZ USER.SSZ.SSHD2.HOSTKEY UACC(NONE)
```

Create resource for token SSZ.TSTUSR.SSH2 for storing private key objects for user tstusr for user public key authentication.

```
RDEFINE CRYPTOZ SO.SSZ.TSTUSR.SSH2 UACC(NONE)
RDEFINE CRYPTOZ USER.SSZ.TSTUSR.SSH2 UACC(NONE)
```

7.3.2 Creating PKCS11 Token

Each PKCS11 token creation/delete is protected by resource SO.token-name in SAF CRYPTOZ class.

Create SSZ.SSHD2.HOSTKEY Token

```
Grant update access to user SSZKMGR for resource SO.SSZ.SSHD2.HOSTKEY
PERMIT SO.SSZ.SSHD2.HOSTKEY CLASS(CRYPTOZ) ID(SSZKMGR) ACCESS(UPDATE)

Logon to TSO as SSZKMGR and goto ICSF Token Management panel

CSFTBR00 ------------- ICSF Token Management - Main Menu -----
OPTION ===> 1

Enter the number of the desired option.

  1  Create a new token
  2  Delete an existing token
```



```
  3  Manage an existing token
  4  List existing tokens

  Full or partial token name: SSZ.SSHD2.HOSTKEY
```

Use option 4 to verify the token is successfully created. Use option 2 to delete an existing token.

Note

After the operation completes, the access can be removed by **PERMIT SO.SSZ.SSHD2.HOSTKEY CLASS(CRYPTOZ) ID(SSZKMGR) DELETE**.

Create SSZ.TSTUSR.SSH2 Token

```
Grant update access to user SSZKMGR for resource USER.SSZ.TSTUSR.SSH2
PERMIT SO.SSZ.TSTUSR.SSH2 CLASS(CRYPTOZ) ID(SSZKMGR) ACCESS(UPDATE)

Logon to TSO as SSZKMGR and goto ICSF Token Management panel

CSFTBR00 ------------- ICSF Token Management - Main Menu -----
OPTION ===> 1

Enter the number of the desired option.

  1  Create a new token
  2  Delete an existing token
  3  Manage an existing token
  4  List existing tokens

  Full or partial token name: SSZ.TSTUSR.SSH2
```

Use option 4 to verify the token is successfully created. Use option 2 to delete an existing token.

Note

After the operation completes, the access can be removed by **PERMIT SO.SSZ.TSTUSR.SSH2 CLASS(CRYPTOZ) ID(SSZKMGR) DELETE**.

7.4 Creating z/OS ICSF PKCS11 Object

Each PKCS11 object is stored in a PKCS11 token. PKCS11 object is protected by resource USER.token-name in SAF CRYPTOZ class.

7.4.1 Creating PKCS11 Object

Create private key objects in token SSZ.TSTUSR.SSH2.

```
Grant update access to user TSTUSR for resource USER.SSZ.TSTUSR.SSH2
PERMIT USER.SSZ.TSTUSR.SSH2 CLASS(CRYPTOZ) ID(TSTUSR) ACCESS(UPDATE)

Logon to z/OS USS as TSTUSR and issue ssh-keygen-g3 command
```

```
$ /opt/tectia/bin/ssh-keygen-g3 -t rsa -b 2048 pkcs11:SSZ.TSTUSR.SSH2
Generating 2048-bit rsa key pair

Key generated.
2048-bit rsa, TSTUSR@S0W1, Wed Nov 26 2025 11:10:55 +0800
Private key saved to PKCS11:SSZ.TSTUSR.SSH2/000000B6
```

```
$ /opt/tectia/bin/ssh-keygen-g3 -t ecdsa -b 384 pkcs11:SSZ.TSTUSR.SSH2
Generating 384 bit ECDSA key on nistp384 curve

Key generated.
384-bit ecdsa, TSTUSR@S0W1, Wed Nov 26 2025 11:10:55 +0800
Private key saved to PKCS11:SSZ.TSTUSR.SSH2/000000B7
```

Note

After the operation completes, the access can be removed by **PERMIT USER.SSZ.TSTUSR.SSH2 CLASS(CRYPTOZ) ID(TSTUSR) DELETE**.

Create private key objects in token SSZ.SSHD2.HOSTKEY. As started task user sshd2 normally does not have authority to logon TSO, user SSZKMGR should be appointed to issue ssh-keygen-g3 command.

```
Grant update access to user SSZKMGR for resource USER.SSZ.SSHD2.HOSTKEY.
PERMIT USER.SSZ.SSHD2.HOSTKEY CLASS(CRYPTOZ) ID(SSZKMGR) ACCESS(UPDATE)

Logon to z/OS USS as SSZKMGR and issue ssh-keygen-g3 command
```

```
$ /opt/tectia/bin/ssh-keygen-g3 -t rsa -b 2048 pkcs11:SSZ.SSHD2.HOSTKEY
Generating 2048-bit rsa key pair

Key generated.
2048-bit rsa, SSZKMGR@S0W1, Wed Nov 27 2025 11:10:55 +0800
Private key saved to PKCS11:SSZ.SSHD2.HOSTKEY/000000B8
```

```
$ /opt/tectia/bin/ssh-keygen-g3 -t ecdsa -b 384 pkcs11:SSZ.SSHD2.HOSTKEY
Generating 384 bit ECDSA key on nistp384 curve

Key generated.
384-bit ecdsa, SSZKMGR@S0W1, Wed Nov 27 2025 11:10:55 +0800
Private key saved to PKCS11:SSZ.SSHD2.HOSTKEY/000000B9
```

Note

After the operation completes, the access can be removed by **PERMIT USER.SSZ.SSHD2.HOSTKEY CLASS(CRYPTOZ) ID(SSZKMGR) DELETE**.

7.5 Using z/OS ICSF PKCS11 Object

PKCS11 object stored in PKCS11 token can be referred by Tectia for z/OS programs using syntax PKCS11:token_name/object_id in filename options. User using object in pkcs11 token must have read access to SAF resource USER.token-name in class CRYPTOZ.

7.5.1 Use PKCS11 Objects in Tectia for z/OS programs

Use PKCS11 private key object as hostkey in program sshd2

```
Grant read access to user SSHD2 for resource USER.SSZ.SSHD2.HOSTKEY
PERMIT USER.SSZ.SSHD2.HOSTKEY CLASS(CRYPTOZ) ID(SSHD2) ACCESS(READ)

Add HostKeyFile statement to /opt/tectia/etc/sshd2_config
HostKeyFile PKCS11:SSZ.SSHD2.HOSTKEY/000000B8
```

Use PKCS11 private key object as user public key authentication in program sshg3, sftpg3 executed by user TSTUSR

```
Grant read access to user TSTUSR for resource USER.SSZ.TSTUSR.SSH2
PERMIT USER.SSZ.TSTUSR.SSH2 CLASS(CRYPTOZ) ID(TSTUSR) ACCESS(READ)

$ /opt/tectia/bin/sshg3 -K PKCS11:SSZ.TSTUSR.SSH2/000000B6 remote_userid@remote_host

$ /opt/tectia/bin/sftpg3 -K PKCS11:SSZ.TSTUSR.SSH2/000000B7 remote_userid@remote_host
```

```
Grant read access to user TSTUSR for resource USER.SSZ.TSTUSR.SSH2
PERMIT USER.SSZ.TSTUSR.SSH2 CLASS(CRYPTOZ) ID(TSTUSR) ACCESS(READ)

add statement 'IdKey PKCS11:SSZ.TSTUSR.SSH2/000000B6' to identificiation file
in .ssh2 directory of user TSTUSR's home directory.

$ /opt/tectia/bin/sshg3 remote_userid@remote_host

$ /opt/tectia/bin/sftpg3 remote_userid@remote_host
```

7.6 Deriving public key from z/OS ICSF PKCS11 Object

ssh-keygen-g3 program can be used to derive public key from z/OS ICSF PKCS11 private key object and stored in .ssh2 directory in user's home directory.

7.6.1 Derive public key from z/OS ICSF PKCS11 Object

ssh-keygen-g3 -D can be used to derive public key from z/OS ICSF PKCS11 private key object and saves to .ssh2 directory of user's home directory


```
Grant read access to user TSTUSR for resource USER.SSZ.TSTUSR.SSH2
PERMIT USER.SSZ.TSTUSR.SSH2 CLASS(CRYPTOZ) ID(TSTUSR) ACCESS(READ)

$ /opt/tectia/bin/sshkey-gen-g3 -D PKCS11:SSZ.TSTUSR.SSH2/000000B6
Public key saved to /u/tstusr/.ssh2/PKCS11-SSZ_TSTUSR_SSH2-000000B6.pub
```


Appendix A Introduction to USS (UNIX)

Tectia Server 6.8 for IBM z/OS is installed to z/OS Unix System Services (USS), which is a certified UNIX operating system, optimized for mainframe architecture, included in z/OS.

The purpose of this appendix is to provide a quick reference to basic information about USS/UNIX for those readers who are not previously familiar with it.

A.1 UNIX File System

The UNIX file system is:

- A data structure or a collection of files
- A hierarchical directory tree with the root ("/") at the top
- Called HFS or zFS
- Actually mounted on a data set

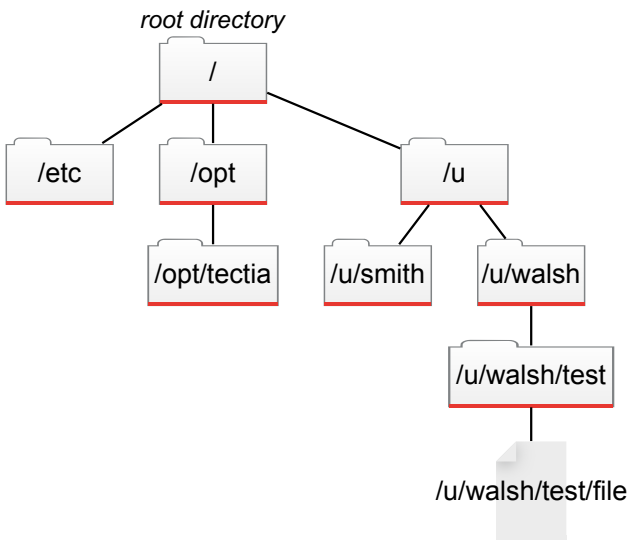

Figure A.1. Hierarchical file system

A.2 UNIX Files vs. MVS Data Sets

You can think of UNIX files like data sets, except that the segments are delimited by '/' (forward slashes) in UNIX, while data sets are delimited by '.' (periods).

Example

Data Set:	`WALSH.TEST.FILE`
UNIX file:	`walsh/test/file`

Note that while the names are similar, these two files do not point to the same location because one is in the UNIX file system while the other one is in the MVS file system.

Another important distinction between UNIX and MVS is that most commands and file names are case-sensitive in UNIX, meaning that a file called `myfile` is different from a file called `MYFILE`.

A.3 Referring to Data Sets

In shell commands (for example when using **scpg3**), MVS data set names must be placed in regular quotation marks (" "). This is to prevent the single quotes and parentheses in the data set names from being interpreted by the shell. Alternatively, you can use backslashes (\) to escape the single quotes and parentheses, for example `//\'USER1.DATASET.NAME1\'` or `//DATASET.NAME1\(MEMBER1\)`.

A.4 Setting Environment Variables in UNIX

If you want to use the Tectia file transfer clients (**sftpg3** and **scpg3**), set the required environment variables either globally (for all users) in `/etc/environment` or per user in `$HOME/.ssh2/environment`.

The simplest way to do this is to copy the contents of `SSHENV` located in `/opt/tectia/doc/zOS/SAMPLIB/` to your environment file. To copy to your user-specific environment file, enter:

```
> cp /opt/tectia/doc/zOS/SAMPLIB/SSHENV ~/.ssh2/environment
```

If you want to use **sshg3** (requiring the use of shell), set the environment variables included in `/opt/tectia/doc/zOS/SAMPLIB/sshsetenv` in the user-specific `$HOME/.profile` (or in any other environment variable file you use).

A.5 Entering USS

When you work through the examples provided in this guide, to get to USS on z/OS, we recommend you to launch the shell environment OMVS directly from TSO, instead of using the ISPF option 6.

To do so when you first log on using a 3270 emulator, first enter **=x** to exit ISPF:

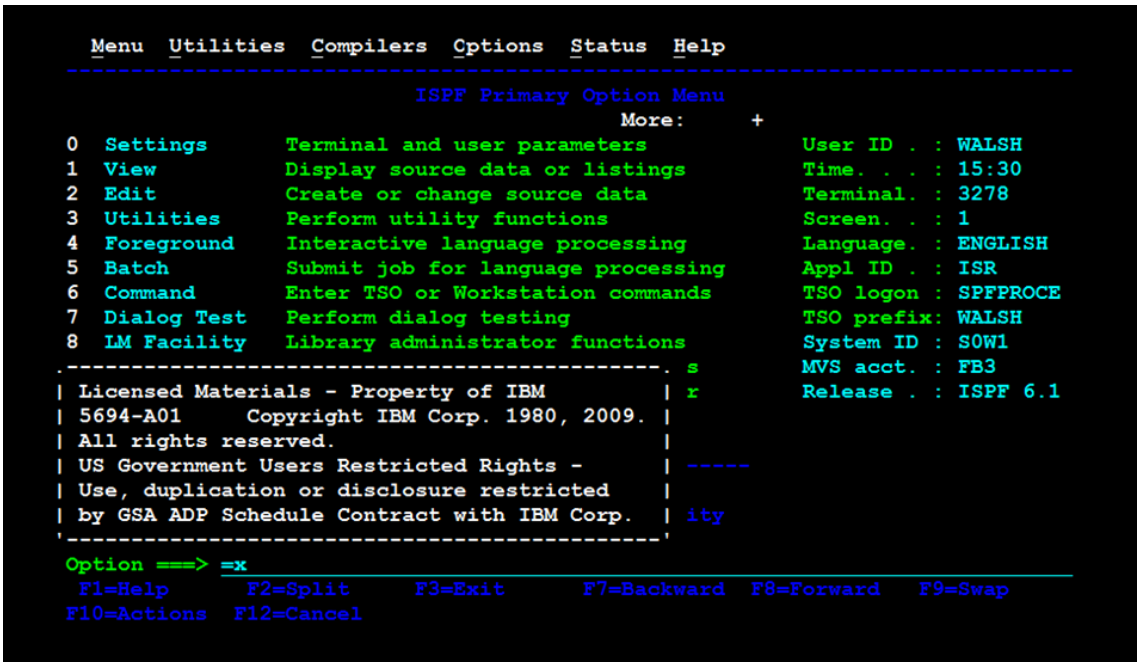

Figure A.2. ISPF

When you are at the TSO `READY` command prompt, type **omvs** and press **Enter**.

Figure A.3. Entering OMVS

You are now in USS (UNIX for z/OS). Take note of the numbers and commands at the bottom of the screen. They are commands that can be executed via the corresponding Function keys. **F6** (TSO) is of particular interest. With this key you can type in TSO commands such as `ISPF 3.4` and press **F6** (do *not* press **Enter** after it) to run the TSO command.

Now, when you exit out of ISPF, you will be taken to the same OMVS session you were running earlier.

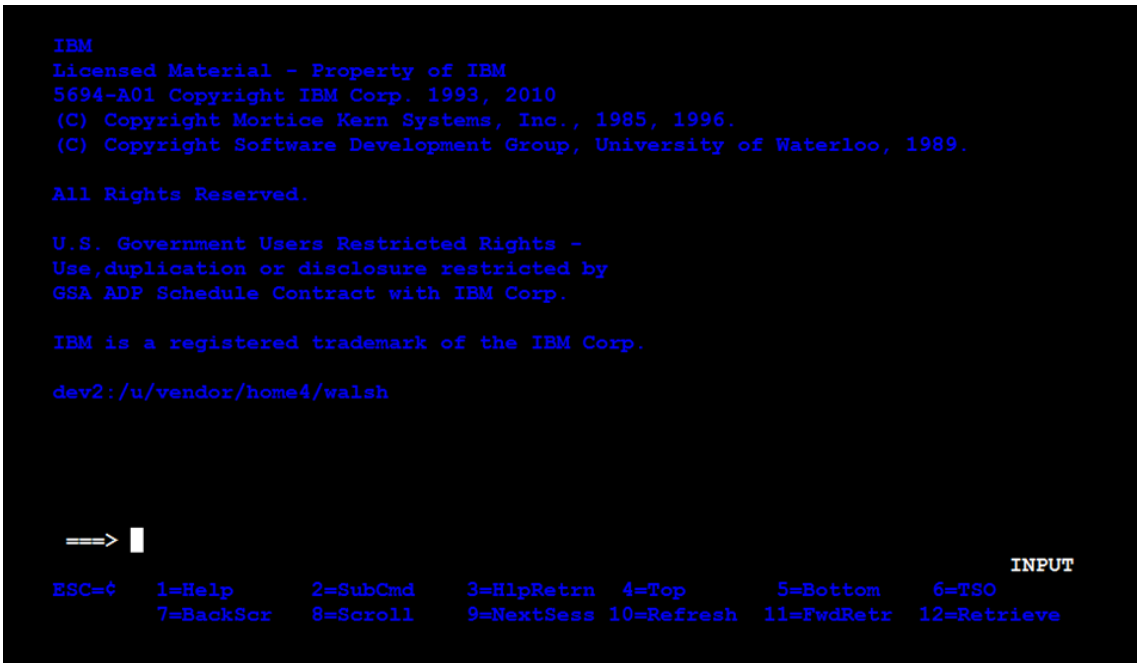

Figure A.4. USS

A.6 File and Directory Permissions in UNIX

File and directory permissions control the ability of users to view and/or make changes to the files and directories in the file system. In UNIX, there are three types of access modes:

- **`read [r]`**: User may look at the file or make a copy of it.
- **`write [w]`**: User may modify or remove the file, or files in a directory.
- **`execute [x]`**: User may execute the file if it is executable.

Access modes are specified for each file and directory three times, for the following distinct classes:

- **`owner`**: The owner of the file or directory
- **`group`**: The group that owns the file or directory
- **`other`**: The other users who do not own the file or directory or belong to the owning group

For example, in `-rwxr-xr-x`

- The first character indicates the file type, which in this case (`-`) is a regular file. (Directories are specified with a `d`.)
- `rwx` indicates that the `owner` of the file has full (read, write and execute) permissions to the file.

- `r-x` indicates that user `group` is allowed to read and execute the file.
- `r-x` indicates that `other` users are allowed to read and execute the file.

File permissions can also be expressed in octal (base-8) notation, which consists of three digits. The first digit specifies the permissions given to the `owner` of the file, the second digit specifies the permissions for the user `group` associated with the file, and the last digit specifies the permissions given to all `other` users.

Table A.1. Permission bits in octal and symbolic notation

Octal notation	Symbolic notation	Meaning
0	`---`	No access
1	`--x`	Execute-only
2	`-w-`	Write-only
3	`-wx`	Write and execute
4	`r--`	Read-only
5	`r-x`	Read and execute
6	`rw-`	Read and write
7	`rwx`	Read, write and execute

For example, `755` (equivalent to `-rwxr-xr-x`) specifies that the owner of the file has full permissions to the file, and the user group and others are allowed to read and execute the file.

`700` (equivalent to `-rwx------`) specifies that the owner of the file has full permissions to the file, and the user group and others do not have access to the file.

To see the permissions of a file in USS, enter the following:

```
> ls -l filename
```

The **ls -l** command lists files in the long format, showing their file type, permissions, number of hard links, file owner, group, file size, and the date of last modification. If you do not specify a file name, the command lists the information for all the files in your current working directory.

A.7 UID

In UNIX, your user identifier, UID, is the numerical representation of your user account. This number was assigned to your account by the person who created your account.

A UID is typically a number between 0 and 65,535. Any account with UID=0 is a superuser account.

To determine your UID, enter **id** in USS.

A.8 MVS vs. UNIX Functional Comparison

Table A.2. Functional comparison between MVS and UNIX

Function	MVS	UNIX
Background work	Submit batch JCL	sh_cmd &
Change working directory	ISPF 3.4	cd
Change local working directory (during FTP or SSH transfer in interactive mode)	ISPF 3.4	lcd
Change permissions	PERMIT	chmod
Configuration parameters	SYS1.PARMLIB	/etc
Data management	DFSMS, HSM	tar, cpio, pax
Debug	TSO TEST	dbx
Delete file or directory	ISPF 3.2 D	rm
Editor	ISPF 2	ed, sed, oedit, ishell
Initiate new task	ATTACH, LINK, XCTL	fork(), spawn()
Interactive access	Logon to TSO	telnet/rlogin to sh/tcsh
Job management	SDSF	ps, kill
List files	ISPF 3.4, LISTC	ls
List user ID attributes	LU	id
Long running work	Started task (STC)	daemon
Make directory	ISPF 3.2 A PDS	mkdir
Post IPL commands	COMMNDxx	/etc/rc
Power user	RACF OPERATIONS	superuser or root
Primary configuration	IEASYSxx	BPXPRMxx
Primary data index	Master Catalog	root (“/”) directory
Procedural language	CLIST, REXX	shell scripts, REXX
Program products	LNKLST	/usr
Resident programs	LPA	sticky bit
System logging	SYSLOG	SYSLOGD
System programs	LNKLST	/bin
Test programs	STEPLIB	/sbin
User data	&SYSUID or &SYSPREF	/u/<*username*>
User identity	user/group	UID/GID

A.9 Further Information

For more detailed information on UNIX, see:

- IBM book *UNIX System Services User's Guide*
- IBM Redbook *UNIX System Services z/OS Implementation*

Index

Symbols

$HOME/.ssh2/environment, 37
/etc/environment, 37

A

advice string, 11

B

batch JCL, 15
BPXBATCH, 15

C

case-sensitivity, 36
ciphers, 23
configuring SOCKS Proxy, 7
cryptographic hardware, 23
  configuring, 23
  optimizing performance, 25
  verifying, 24

D

data set names, 36

E

enabling cryptographic hardware, 24
entering USS, 37
environment variables in UNIX, 37

F

file and directory permissions in UNIX, 39
file transfer advisor, 11
file transfer examples, 15
file transfer profiles, 12, 21
  filename-matched, 13
  named, 13
filename-matched file transfer profiles, 13
files vs. data sets, 36
filetype JES, 19
FTP-SFTP conversion, 7
  important files, 7
  setting up, 7

H

hierarchical file system, 35

I

ICSF PKCS11 Token Setup, 27

J

JCL, 15
  with filetype=JES, 19
JCL jobs (managing over SFTP)
  deleting, 20
  fetching, 20
  listing, 20
  submitting, 19
JES, 19

M

MACs, 23
MVS vs. UNIX functional comparison, 41

N

named file transfer profiles, 13

O

octal notation, 40

P

permission bits in UNIX, 40
permissions in UNIX, 39

Q

quoting, 36

R

referring to data sets, 36

S

SAMPLIB, 15
scpg3, 15
secure file transfer, 15
setting up cryptographic hardware, 23
sftpg3, 15
SOCKS Proxy, 7
  configuration file, 7
  configuring, 7
  running, 9
ssh-socks-proxy, 9
ssh-socks-proxy-config.xml, 7
SSHENV, 37

T

TSO commands, 38

U

UID, 40
UNIX
  environment variables, 37
  file and directory permissions, 39
  file system, 35
  functions, 41
  permission bits, 40
  quick reference, 35
user identifier, 40
USS, 35
  entering, 37
  file system, 35

V

verifying cryptographic hardware use, 24

Z

z/OS ICSF PKCS11 Introduction, 27
z/OS ICSF PKCS11 Token Dataset, 28