Co:Z® Co-Processing Toolkit for z/OS

Co:Z SFTP - User's Guide

V 5.1.1 Edition

Published June, 2018

Table of Contents

1. Introduction

z/OS OpenSSH and IBM Ported Tools OpenSSH (for z/OS versions older than V2R2) include a port of the popular OpenSSH tools. These tools provide for secure remote login and program execution (**ssh**) and file transfer (**sftp** and **scp**). The **sftp** and **sftp-server** commands implement a file transfer program that is similar to ftp, but use ssh for their underlying secure transport. The sftp specification [1] accounts only for binary transfers. The IBM z/OS sftp client has been enhanced to support ASCII-EBCDIC conversion. Dataset support is not provided.

The Co:Z Co-Processing Toolkit for z/OS includes Co:Z SFTP - a port of the OpenSSH (v6.4p1) **sftp-server** subsystem and **sftp** command (renamed as **cozsftp**). Extensive enhancements have been made to support z/OS facilities such as z/OSS datasets and spool files. z/OS OpenSSH or IBM Ported Tools OpenSSH, as applicable based on the z/OS version, is required since Co:Z does not provide the base ssh and sshd components.

1.1 Features

- Co:Z is compatible with most existing sftp products (see *the section called "Client Requirements"*).
- Transfer datasets via the **get** and **put** commands
- Navigate z/OS catalogs via the **cd** command.
- List dataset information and PDS directories via the **ls** command.
- Records SMF 119 records if user has `BPX.SMF` SAF authorization.
- Support for IBM FTP compatible user exits. A guide to setting up and using these exits can be found here: *http://dovetail.com/docs/sftp/coz_sftp_exits.pdf*.
- Supports direct access to datasets which can be opened in sequential, record mode by the `fopen()` C-library routine. This includes:
  - MVS sequential datasets (QSAM, BSAM, VSAM)
  - PDS and PDSE members
  - SYSOUT datasets, including the MVS internal reader
- Supports JES2 and JES3 job submission, status and spool file transfer on z/OS 1.9 or later. Future releases of Co:Z SFTP will also support cancel and purge facilities.
- Supports text or binary conversion via flexible line-termination rules:
  - Cr, Lf/Newline, CrLf, Cr and/or Lf, RDW, none
- Supports flexible record padding / overflow rules
- Can specify dataset dynamic allocation (BPXWDYN) keywords

[1] SFTP specification: *http://tools.ietf.org/html/draft-ietf-secsh-filexfer-02*

- Can specify name patterns to automatically associate file transfer options to POSIX files and datasets

1.2 Supported Environments

z/OS Host Requirements

- V2R1 or later, with a minimum machine architecture of z10 and above

Note

Users running machine architectures lower than z10 or a z/OS release lower than V2R1 should use Co:Z release 4.5.1.

- *z/OS OpenSSH* or *IBM Ported Tools OpenSSH*

  z/OS V2R2 includes OpenSSH. Earlier versions of z/OS require IBM Ported Tools OpenSSH v1.2 (or later) to be installed. See the version of our *Quick Install Guides* matching your z/OS OpenSSH version for additional information.

Client Requirements

- Co:Z SFTP is compatible with a wide variety of operating systems, including Windows, *IX variants, z/OS...
- Products supporting the SSH File Transfer Protocol, such as:
  - OpenSSH
  - puTTy psftp
  - winSCP
  - gFTP
  - Many commercial implementations.

  Note: Not all products support all of the Co:Z SFTP extensions. Refer to *Client Compatibility* for additional information.

2. Co:Z SFTP Configuration

In order to use Co:Z SFTP, installation is required for the *Co:Z Toolkit for z/OS*. Be sure to make note of the installation directory.

You do *not* need to install the Co:Z Target System Toolkit on your remote systems to use Co:Z SFTP. A compatible SSH/SFTP product is all that is required.

2.1 Co:Z SFTP Quick Start

After completing the installation of the *Co:Z Toolkit for z/OS*, the following are the minimum steps to get started using Co:Z SFTP. For more detailed information, see the remaining chapters in this guide.

On z/OS:

1. Edit `/etc/ssh/sshd_config`. Comment out the existing sftp subsystem line and add the following:

```
Subsystem sftp <COZ_INST>/bin/sftp-server.sh
```

   Restart SSHD by executing:

```
kill -HUP `cat /var/run/sshd.pid`
```

2. Copy the site-wide sample configuration files to `/etc/ssh`:

```
cp <COZ_INST>/samples/sftp-server.site.rc /etc/ssh/sftp-server.rc
chmod 755 /etc/ssh/sftp-server.rc

cp <COZ_INST>/samples/cozsftp_server_site_config /etc/ssh/cozsftp_server_config
chmod 644 /etc/ssh/cozsftp_server_config

cp <COZ_INST>/samples/cozsftp_site_config /etc/ssh/cozsftp_config
chmod 644 /etc/ssh/cozsftp_config
```

3. Edit `/etc/ssh/sftp-server.rc` and uncomment `USE_COZ_SFTP=true` to enable Co:Z SFTP for all sftp users.

4. Using an SSH connection to z/OS (i.e, PuTTY from Windows or OpenSSH for unix), test a Co:Z SFTP client connection to the Co:Z SFTP Server using 127.0.0.1. Note: TSO OMVS cannot be used for this test because a password prompt does not work in this environment.

```
/u/home/user>cozsftp user@127.0.0.1
Co:Z SFTP version: 4.2.0 (6.4p1) 2017-01-10
Copyright (C) Dovetailed Technologies, LLC. 2008-2017. All rights reserved.
Connecting to 127.0.0.1...
Connected to 127.0.0.1.
```

```
Connection established, local_addr=127.0.0.1 local_port=1345 remote_addr=127.0.0.1 remote
cozsftp> ls /+
/+/error.log    /+/loglevel=I   /+/mode=binary
cozsftp> exit
/u/home/user>
```

The command **ls** /+ is a special Co:Z SFTP command used to set file transfer options. If the response to this command is `Can't ls: "/+" not found`, then the Co:Z SFTP server installation is not correct. Recheck the installation steps to determine the error.

Once you have verified your Co:Z Toolkit installation for Co:Z SFTP, try connecting to the Co:Z SFTP server from a remote system with OpenSSH. Then work through the features described in *Chapter 3, Using the Co:Z SFTP server*. Next try connecting from z/OS using the Co:Z SFTP client command, **cozsftp**, to a remote SFTP server and work through the features described in *Chapter 4, Using the Co:Z SFTP client*.

2.2 Configuring the Co:Z SFTP Server

The configuration discussed here is designed to allow individual users to use either the original sftp-server or the enhanced Co:Z version, depending on their configuration. The default setup makes for an ideal beta testing environment, as only designated users will use the enhanced Co:Z sftp-server.

Modifying the SFTP Subsystem

1. Update the `sshd_config` file, typically located at `/etc/ssh/sshd_config` to modify the sftp subsystem definition: [1]

```
#Subsystem      sftp    /usr/lib/ssh/sftp-server     ❶
Subsystem       sftp    <COZ_INST>//bin/sftp-server.sh     ❷
```

   ❶ The original sftp subsystem line should be commented out.

   ❷ The new subsystem line should point to the `sftp-server.sh` shell script located in the Co:Z installation `bin` directory. This script is designed to run the original sftp-server by default, but will run the Co:Z version if the user has configured it. See *the section called "User specific customization"* for details. The installation process should have marked this file as executable, but this should be verified.

2. If OpenSSH **sshd** was running prior to editing `sshd_config`, it should be reinitialized. This can be done by sending `SIGHUP` to the running process. The pid for this process is typically in the file `/var/run/sshd.pid`:

```
kill -HUP `cat /var/run/sshd.pid`
```

[1] It is sometimes convenient to set up a *test* OpenSSH server where this subsystem can be easily modified. To do this see: *Appendix J, Setting up a test OpenSSH system on z/OS*.

Co:Z SFTP Server configuration overview

The following table describes how a Co:Z SFTP Server session is started and outlines the sequence of configuration steps that occur prior to the establishment of the session. Details on these configuration steps follow the table.

Table 2.1. Co:Z SFTP Server initialization steps

Step	Configuration	Notes
1	$COZ_HOME/bin/sftp-server.sh	This shell script is executed by z/OS OpenSSH **sshd** upon a request for an SFTP server subsystem. *This file should not be modified by the installation*, but you may want to review the comments at the beginning of the script. This script will execute the site-wide and user-specific rc scripts and configuration files (see following steps).
2	/etc/ssh/sftp-server.rc	Site-wide environment variable configuration.
3	$HOME/.ssh/sftp-server.rc	User specific environment variable configuration. Can contain customized log file location, logging and tracing options, etc. The location of this file may be changed by setting the $COZ_SFTP_USER_SERVER_RC environment variable.
4	$HOME/.ssh/cozsftp_server_config	User-specific configuration settings. User customized file patterns may be specified here. File patterns here override those found in the site-wide file below. The location of this file may be changed by setting the $COZ_SFTP_USER_SERVER_CONFIG environment variable.
5	/etc/ssh/cozsftp_server_config	Site-wide configuration settings. Site-wide notification, fixed, default and file pattern settings.

Sitewide server configuration

The Co:Z SFTP Server can be configured with system-wide defaults by creating and configuring the file `/etc/ssh/sftp-server.rc`. A sample file (`sftp-server.site.rc`) is provided in the `<COZ_INST>/samples`, and should be copied to the `/etc/ssh` directory:

```
cp <COZ_INST>/samples/sftp-server.site.rc /etc/ssh/sftp-server.rc
chmod 755 /etc/ssh/sftp-server.rc
```

Sample site sftp-server.rc file

```
#!/bin/sh
# Set site-wide environment variables for Co:Z SFTP server.
# Place this sample as an executable script in file: /etc/ssh/sftp-server.rc

# Uncomment the following to make CO:Z SFTP the default for all users
#USE_COZ_SFTP=true ❶

# The following are the default locations for user level configuration files.
#COZ_SFTP_USER_SERVER_RC=$HOME/.ssh/sftp-server.rc ❷
#COZ_SFTP_USER_SERVER_CONFIG=$HOME/.ssh/cozsftp_server_config ❸
```

❶ By default, the **sftp-server.sh** script discussed above will execute the IBM version of sftp-server. The `USE_COZ_SFTP` environment variable can be used to make Co:Z SFTP Server the default for *all* users, even if they don't have their own `sftp-server.rc` file.

❷ In some cases, Co:Z SFTP users may not have access to individual `$HOME` directories or it may be desirable to have all user configuration files centralized. In this case, the environment variable `COZ_SFTP_USER_SERVER_RC` can be specified to provide an alternate file name for the user .rc file in a common, readable location. For example, to specify a common directory for all user configuration files, set the following:
`COZ_SFTP_USER_SERVER_RC=/usr/share/coz/$LOWER_LOGNAME.sftp-server.rc`

To disable the usage of user specific sftp-server.rc files for all users, `COZ_SFTP_USER_SERVER_RC` can be set to a dummy file name (e.g: `/dummy`); however, this requires that `USE_COZ_SFTP` be set to `true` in order to activate Co:Z SFTP for all users.

Note that the z/OS Unix System Services `$LOGNAME` environment variable holding the current username is in uppercase. As this is not always consistent with other POSIX style usage, the **sftp-server.sh** script exports an enviroment variable named `$LOWER_LOGNAME` that downcases the value in `$LOGNAME`.

❸ Additionally, individual user server config files (where pattern based file transfer options are set) can be similarly located. To learn more about config files, refer to section: *Appendix C, Session config files*. By default, user server config files are located at `$HOME/.ssh/cozsftp_server_config`.

Note: The `/etc/ssh/sftp-server.rc`, if present, must be marked executable, as must the individual user files.

System-wide defaults for customizing options available for Co:Z SFTP server sessions can be configured by creating and configuring the file `/etc/ssh/cozsftp_server_config`. A sample file (`cozsftp_server_site_config`) is provided in the `<COZ_INST>/samples`, and should be copied to the `/etc/ssh` directory:

```
cp <COZ_INST>/samples/cozsftp_server_site_config /etc/ssh/cozsftp_server_config
chmod 644 /etc/ssh/cozsftp_server_config
```

For information on the session options available for configuration, see *Appendix C, Session config files*.

Restricting OpenSSH users to SFTP

Some installations prefer to restrict ssh users to a certain set of commands like the sftp-server, rather than giving

them interactive shell access. See *Appendix I, Restricting OpenSSH users to SFTP* for a technique to enforce this restriction.

User specific customization

By default, the **sftp-server.sh** script discussed above will execute the IBM version of sftp-server. Individual users can activate the Co:Z version of `stfp-server` by creating a profile script, `sftp-server.rc`, in their home .ssh directory:

```
# if the user's .ssh does not exist:
mkdir $HOME/.ssh
chmod 700 $HOME/.ssh

cp <COZ_INST>/samples/sftp-server.user.rc $HOME/.ssh/sftp-server.rc
chmod u+x $HOME/.ssh/sftp-server.rc
```

Note: Removing or renaming this file will re-enable the IBM version of sftp-server, unless `USE_COZ_SFTP=true` has been set by the site.

Sample user sftp-server.rc file

```
#!/bin/sh
# The presence of this executable script in $HOME/.ssh/sftp-server.rc
# will cause the COZ version of sftp-server to be used

# You may uncomment and set the following options to override the defaults:
#export SFTP_ZOS_OPTIONS="mode=text" ❶
#export SFTP_ZOS_INITIAL_DIR=// ❷
#export SFTP_LOGFILE=$HOME/sftp.log ❸

# The Co:Z support team may request that you uncomment the following options
# to enable tracing:
#export SFTP_SERVER_OPTIONS="-e -l debug3"
#export COZ_LOG=T
```

❶ The `SFTP_ZOS_OPTIONS` environment variable can be used to set the default options for the user. Multiple options may be specified, separated by commas. The options are described here: *Appendix B, Co:Z SFTP options*.

❷ The `SFTP_ZOS_INITIAL_DIR` environment variable can be used to override the home directory on the server. By default this is the user's USS home directory. If the string `//` or `/-/` is supplied, the user's MVS top level qualifier is used. Otherwise an absolute path (USS or MVS dataset space) may be supplied.

❸ Log files are created for every sftp server session; these files are of particular interest in case a problem is encountered and additional error detail is needed. See *the section called "Co:Z SFTP Server logging"* for additional information.

User specific defaults for customizing options available for Co:Z SFTP server sessions can be configured by creating and configuring the file `/etc/ssh/cozsftp_server_config`. A sample file (`cozsftp_server_user_config`) is provided in the `<COZ_INST>/samples`, and can be copied to the user's `.ssh` directory:

```
cp <COZ_INST>/samples/cozsftp_server_user_config $HOME/.ssh/cozsftp_server_config
chmod 644 $HOME/.ssh/cozsftp_server_config
```

For information on the session options available for configuration, see *Appendix C, Session config files*.

Co:Z SFTP Server logging

Log files are created for every sftp server session; these files are of particular interest when a problem is encountered and additional error detail is needed. The following sections define how to control the logging destination as well as logging levels.

Logging Destination

Logging may be directed to the filesystem (/tmp by default) or to SYSOUT:

- Filesystem

  By default, log files are written to the `/tmp` directory (or the directory specified by the `TMPDIR` environment variable, if it is set). To change this default for all users, modify `/etc/ssh/sftp-server.rc` as needed. Individual users can override this setting by exporting `SFTP_LOGFILE` in the copy of `sftp-server.rc` in their individual `.ssh` directory.

  The directory containing the log files must be cleaned up and monitored for space; however, it is important to keep these files for some period of time in order to allow:

  - the current session log file to be accessed by the remote sftp client (e.g: get /+error.log) to get details of a problem, and
  - support personnel to review the session log file for diagnostic information when investigating a problem.

  In many cases, installations will choose to put Co:Z SFTP server session logs in a separate zFS or HFS filesystem. See the *IBM Ported Tools OpenSSH v1.3 / z/OS V2R2 OpenSSH - Quick Install Guide* for additional information on managing the /tmp filesystem.

- SYSOUT

  Optionally, logging output may be redirected to a SYSOUT spool file. To enable this, update the `/etc/ssh/sftp-server.rc` or user-specific `$HOME/.ssh/sftp-server.rc` script as follows:

```
SFTP_LOG_SYSOUT=true        # required
SFTP_SYSOUT_CLASS=H         # optional
unset SFTP_LOGFILE          # don't set this
export _BPX_JOBNAME=COZLOG  # recommended
```

  If this feature is enabled:

  - an additional OMVS address space will be created for each connection to write the log, and

- remote SFTP connections will not be able to get the current session log file using the "get /+error.log" command.

SDSF can be used to locate a user's logfile when needed for problem diagnosis. When the session is active, the output will be displayed by SDSF.DA. Once the session has ended, output is available in SDSF.H or SDSF.O, depending on whether the spool class/file is held. The jobid assigned to the output is from a pool of OMVS started tasks; therefore, is not unique and not owned by the SFTP user. The output can be identified by the jobname and the creation date. The following SDSF commands are useful:

- `arr crdate 20` - expands the CrDate field to show the time as well as the date
- `sort crdate d` - sorts descending by date/time

If you would like to download the log spool file using Co:Z SFTP, first find the job in SDSF. Next, use the "?" prefix command to find the DSID. Finally, use the following commands in a remote SFTP client to download it:

```
ls /+mode=text
get //-JES/STCxxxxx/nnn logfile.txt
```

Note

Directing the Co:Z SFTP server log file to `/dev/console` is *NOT* recommended for the following reasons:

- the remote client will not be able to retrieve the log via the special /+error.log file name,
- when enabling debug logging to troubleshoot a problem, a very large amount of data will be written to the z/OS console, and
- when log messages go to the z/OS console, it is difficult to collect messages for a particular session which significantly impacts problem diagnosis.

If console messages are needed for automation, see *Chapter 5, Automation with System Console Messages* for additional information.

Logging Level

The logging level is controlled by exporting the `COZ_LOG` and/or `SFTP_SERVER_OPTIONS` variables. To set these variables for all users, modify the `/etc/ssh/sftp-server.rc` as needed. Individual users can override these setting by exporting the variables in a copy of `sftp-server.rc` in their individual `.ssh` directory. Additionally, the Co:Z log level can be set with the Co:Z SFTP `loglevel` option. See *Section B.2, "Miscellaneous options"* for additional information.

- `SFTP_SERVER_OPTIONS` allows command line options to be set for the sftp-server. The default is `"-e -l info"` which is required in order to route messages to `SFTP_LOGFILE`. `"-e -l debug3"` may be used to configure debug-level logging in sftp-server code.
- `COZ_LOG` controls logging options for the Co:Z extension library used to add z/OS support to sftp-server. The

default is `I` which logs error, warning and informational messages to `SFTP_LOGFILE`. This variable may be set to one of E, W, N, I, D, T or F for Error, Warning, Notice, Informational, Debug, Trace, or Fine logging levels as requested by the Co:Z support team.

2.3 Configuring the Co:Z SFTP Client

Client configuration overview

The following table describes how a Co:Z SFTP client (cozsftp) session is started and outlines the sequence of configuration steps that occur prior to the establishment of the session. Details on these configuration steps follow the table.

Table 2.2. Co:Z SFTP Client initialization steps

Step	Configuration	Notes
1	$COZ_HOME/bin/cozsftp	This shell script is is executed either interactively by a z/OS user or in a batch job. *This file should not be modified by the installation*, but you may want to review the comments at the beginning of the script. This script will execute the site-wide and user-specific rc scripts and configuration files (see following steps).
2	/etc/ssh/cozsftp_client.rc	Site-wide environment variable configuration.
3	$HOME/.ssh/cozsftp_client.rc	User specific environment variable configuration. Can contain customized options to the sftp command itself and/or custom logging settings. The location of this file may be changed by setting the $COZ_SFTP_USER_RC environment variable.
4	$HOME/.ssh/cozsftp_config	User-specific configuration settings. User customized file patterns may be specified here. File patterns here override those found in the site-wide file below. The location of this file may be changed by setting the $COZ_SFTP_USER_CONFIG environment variable.
5	/etc/ssh/cozsftp_config	Site-wide configuration settings. Site-wide notification, fixed, default and file pattern settings.

Sitewide client configuration

The **cozsftp** client command can be configured with system-wide defaults by creating and customizing the file `/etc/ssh/cozsftp_client.rc`. A sample file (`cozsftp_client.site.rc`) is provided in the `<COZ_INST>/samples`, and may be copied to the `/etc/ssh` directory:

```
cp <COZ_INST>/samples/cozsftp_client.site.rc /etc/ssh/cozsftp_client.rc
chmod 755 /etc/ssh/cozsftp_client.rc
```

Sample site cozsftp_client.rc file

```
#!/bin/sh
# Set site-wide environment variables for Co:Z SFTP client.
# Place this sample as an executable script in file: /etc/ssh/cozsftp_client.rc

# Uncomment the following to set command line options for the cozsftp command
#COZSFTP_CLIENT_OPTS=  ❶

# The following are the default locations for user level configuration files.
#COZ_SFTP_USER_RC=$HOME/.ssh/cozsftp_client.rc ❷
#COZ_SFTP_USER_CONFIG=$HOME/.ssh/cozsftp_config ❸
```

❶ This environment variable can be used to specify site-wide **cozsftp** command line options.

❷ In some cases, Co:Z SFTP users may not have access to individual `$HOME` directories or it may be desirable to have all user configuration files centralized. In this case, the environment variable `COZ_SFTP_USER_RC` can be specified to provide an alternate location for individual .rc files in a common, readable location. For example, to specify a common directory for all user configuration files, set the following: `COZ_SFTP_USER_RC=/usr/share/coz/$LOWER_LOGNAME.cozsftp_client.rc`

Note that the z/OS Unix System Services `$LOGNAME` environment variable holding the current username is in uppercase. As this is not always consistent with other POSIX style usage, the **sftp-server.sh** script exports an enviroment variable named `$LOWER_LOGNAME` that downcases the value in `$LOGNAME`.

❸ Additionally, individual user client config files (where pattern based file transfer options are set) can be similarly located. To learn more about config files, refer to section: *Appendix C, Session config files*. By default, user client config files are located at `$HOME/.ssh/cozsftp_config`.

Note: The `/etc/ssh/cozsftp_client.rc`, if present, must be marked executable, as must the individual user files.

System-wide defaults for customizing options available for Co:Z SFTP client sessions can be configured by creating and configuring the file `/etc/ssh/cozsftp_config`. A sample file (`cozsftp_site_config`) is provided in the `<COZ_INST>/samples`, and should be copied to the `/etc/ssh` directory:

```
cp <COZ_INST>/samples/cozsftp_site_config /etc/ssh/cozsftp_config
chmod 644 /etc/ssh/cozsftp_config
```

For information on the session options available for configuration, see *Appendix C, Session config files*.

User specific customization

When the `cozsftp` client command is invoked, the contents of the optional file `$HOME/.ssh/cozsftp_client.rc` are dotted into the environment at the start of the command.

Most users will not require this file, but it may be used to automatically provide *command line arguments* to the cozsftp client without having to explicitly code them every time the client is invoked. The desired command line arguments *must* be made available in the environment variable `COZSFTP_CLIENT_OPTS`.

Sample user cozsftp_client.rc file

```
#!/bin/sh
# Set user-specific environment variables for Co:Z SFTP client.
# Place this sample as an executable script in file: $HOME/.ssh/cozsftp_client.rc

# Uncomment the following to set command line options for the cozsftp command
#   For example, to allow new host keys to be created automatically:
#COZSFTP_CLIENT_OPTS="$COZSFTP_CLIENT_OPTS -oStrictHostKeyChecking=no"
```

User specific defaults for customizing options available for Co:Z SFTP client sessions can be configured by creating and configuring the file `/etc/ssh/cozsftp_config`. A sample file (`cozsftp_user_config`) is provided in the `<COZ_INST>/samples`, and can be copied to the user's `.ssh` directory:

```
cp <COZ_INST>/samples/cozsftp_user_config $HOME/.ssh/cozsftp_config
chmod 644 $HOME/.ssh/cozsftp_config
```

For information on the session options available for configuration, see *Appendix C, Session config files*.

3. Using the Co:Z SFTP server

3.1 Setting, displaying and clearing file transfer options

Unlike standard FTP, SFTP has no **site** command for setting platform specific options. Co:Z SFTP file transfer options are set with a special **ls** command request of the form: **ls /+<name=value>**. They can be cleared with a request of the form: **ls /+NO<name>**.

Multiple options can be set by separating the key=value pairs with commas. An error is returned if one or more of the options was incorrectly specified, but the remaining options are set as requested.

The options directory `/+/` is a pseudo directory on the server, and it is possible to make it the working directory via the **cd** /+ command. From this directory, options may be set and listed without the `/+` prefix.

The active options and their settings can be displayed by issuing the command **ls** /+.

Co:Z SFTP server file transfer options may be specified interactively or via configuration files. The active options are determined in the following priority order:

1. The `fixed:` section of `/etc/ssh/cozsftp_server_config` (highest priority and non-modifiable)
2. The first matching pattern (if any) from `$HOME/.ssh/cozsftp_server_config`
3. The first matching pattern (if any) from `/etc/ssh/cozsftp_server_config`
4. Previous interactive commands: `ls /+` (described below) in the same session
5. The environment variable `SFTP_ZOS_OPTIONS`
6. The `default:` section of `/etc/ssh/cozsftp_server_config` (lowest priority)

For a list of available options, see *Appendix B, Co:Z SFTP options*.

For a description of the `cozsftp_server_config` file format, including how to specify file name patterns, see *Appendix C, Session config files*.

All examples in the following sections can be run by most sftp clients, either from z/OS or from other platforms (Windows, linux, etc..). **Note:** There are some differences in the way clients interact with the server, so the output shown in the examples below (performed with the OpenSSH sftp client) may not match your output exactly.

Example: Setting and displaying basic options

```
sftp> ls /+mode=text     ❶
/+mode=text
sftp> ls /+     ❷
/+/clientcp=iso8859-1            /+/error.log
/+/loglevel=I                    /+/mode=text
/+/servercp=IBM-1047             /+/trim
```

❶ The option command `ls /+mode=text` is used to set the transfer mode to text. **mode=binary** is the default.
❷ The option list command `ls /+` shows the options currently in effect. In this case, the codepages `clientcp` and `servercp` are set to the defaults.

Example: Setting multiple options

```
sftp> ls /+lrecl=80,recfm=fb,space=trk.3.2     ❶
/+lrecl=80,recfm=fb,space=trk.3.2
```

❶ Multiple options can be specified, separated by commas. Note that the SPACE parameter uses periods for commas to avoid ambiguity.

Example: Showing all options

```
sftp> ls /+showall      ❶
/+showall=true
sftp> cd /+     ❷
sftp> ls     ❸
NOblksize             NObufno               clientcp=ISO8859-1
conddisp=catlg        NOcopies              NOdataclas
NOdest                NOdir                 NOdisp
NOdsntype             NOdsorg               error.log
estsize               NOforms               NOgdgnt
NOhold                NOjesjobname          NOjesjobwait
jeslrecl=80           jesowner=KIRK         jesrecfm=f
jesstatus=*           NOlabel               NOlike
linerule=flexible     loglevel=I            lrecl=80
NOmaxvol              NOmgmtclas            mode=text
NOmount               NOnorecall            NOoutdes
overflow=wrap         NOpad                 recfm=fb
NOrelease             replace               NOreqexits
NOreset               NOretpd               NOsequence
servercp=IBM-1047     showall               smf
space=trk.3.2         NOspin                NOstorclas
NOsysout              trim                  NOtrtch
NOucount              NOunit                NOvol
```

```
NOwriter
sftp> ls noshowall,norecfm      ❹
noshowall,norecfm
sftp> ls
clientcp=ISO8859-1    error.log             loglevel=I
lrecl=80              mode=text             servercp=IBM-1047
space=trk.3.2         trim
```

❶ The option command **ls /+showall** is used to set the option listing mode to show all options, even those that are not active.

❷ Since the options are treated as entries in a pseudo directory, the cd command can be used to make that directory the working remote directory.

❸ Issuing the **ls** from the options directory will show all of the options. Those that are not active are prefixed with the string `NO`. Note that the options can be listed even if the current working directory is not the options pseudo dir with the command **ls** /+.

❹ Active options can be de-activated by prefixing the option with the string `NO`. In this example, the `showall` option is cleared, as well as the `recfm` option.

3.2 Reading the error log

Most implementations of the sftp specification, including OpenSSH, do not allow for transmission of detailed information from the server to the client in the event of an error. Adding dataset transfers to the mix only increases the need for better error reporting. To help alleviate this problem, the Co:Z sftp implementation provides a comprehensive logging facility that can be enabled and tuned by each user session.

Several of the above option listing examples show `error.log` as one of the options. This is actually an alias for the running session's log file, which is usually written to the `/tmp` directory (See *Chapter 2, Co:Z SFTP Configuration* for more information on where this file is written). This alias can be used to easily retrieve the log at anytime and examine it from the client.

This feature makes it possible to examine detailed error information from the client without having to abandon the active sftp session. Users of graphical clients such as **winSCP** and **gFTP** see an even greater advantage in that the error.log file can be viewed simply by selecting the file and transferring it in `view` mode.

Example: Getting and displaying the error log

```
sftp> rm //user.coz.sampjcl     ❶
Removing //user.coz.sampjcl
Couldn't delete file: Failure     ❷
sftp> get /+/error.log     ❸
Fetching /+/error.log to error.log
/+/error.log                          100%   68     0.1KB/s   00:00
sftp> !cat error.log     ❹
ZosUtil[E]: Dataset "USER.COZ.SAMPJCL" is a PDS. Use rmdir instead.
sftp>
```

❶ This command attempts to delete a PDS with the **rm** which is not allowed.

❷ The request fails, but the standard sftp error message is not very helpful.

❸ To get better information, the `error.log` from the options directory is requested.
❹ Using the local shell command **cat** to display the log gives detailed error information.

3.3 Working with Datasets

The Co:Z implementation of sftp accepts two prefix strings to identify MVS datasets as absolute paths. The first (`//`) is consistent with IBM's common usage. A secondary form (`/-/`) is also available, as not all sftp clients will allow double slash characters to be sent.

Navigating Datasets

The sftp **cd** command can be used to navigate around the z/OS dataset space. Using the dataset prefix `//` or `/-/`, the dataset space can be entered. Once there, traversal up and down various dataset levels can be performed similarly to hierarchical file systems.

Partitioned datasets are treated as directories as well. Once a PDS is made the current working directory, its members can be listed and retrieved like normal files.

Just as listing the entire catalog from the root is not allowed, it is not possible to make the catalog root the current working directory. As such, the command **cd //** will fail.

Example: Navigating the dataset space

```
sftp> cd //user     ❶
sftp> pwd     ❷
Remote working directory: //USER
sftp> cd coz.testjcl     ❸
sftp> pwd
Remote working directory: //USER.COZ.TESTJCL
sftp> cd ..     ❹
sftp> pwd
Remote working directory: //USER.COZ
```

❶ Using the dataset prefix `//`, the high level qualifier `user` is specified. For cd commands, the dataset name is case insensitive.

❷ The **pwd** command will list the current working dataset level. Note that the name is properly displayed in uppercase

❸ Multiple levels can be traversed at a time. Instead of using the normal separator (`.`), a slash can be used: **cd coz/testjcl**.

❹ The **cd ..** command will move up a level, as expected.

Transferring Datasets

The `get` and `put` commands are used to transfer datasets and PDS members. By default, the transfer mode is binary, and when storing new datasets, the DCB defaults are determined by the system and are often RECFM=U.

Any options previously set via the **ls /+option=value** are in effect for any given transfer.

Note

When using the `put` command to write datasets, the target name is used to determine the actual dataset name written. In most cases this determination is straight forward, but in certain circumstances, name

determination is more involved. See *Appendix D, Dataset Name Determination* for complete details.

Example: Get a text sequential dataset

```
$ sftp user@zos.com     ❶
Connecting to zos.com...
user@zos.com's password:
sftp> ls /+mode=text     ❷
/+mode=text
sftp> get //USER.LOG.MISC     ❸
Fetching //USER.LOG.MISC to USER.LOG.MISC
```

❶ This example shows the full connection process, using keyboard-interactive password authentication.

❷ The default transfer mode of binary is overridden and set to `text`.

❸ The `get` command uses the dataset path prefix `//` (or, optionally `/-/`) to specify that a dataset is being requested.

Example: Get PDS members

```
sftp> ls /+     ❶
/+/clientcp=ISO8859-1     /+/mode=text     /+/servercp=IBM-1047
sftp> get //user.ssh.jcl(sshd)     ❷
Fetching //user.ssh.jcl(sshd) to user.ssh.jcl(sshd)
```

❶ If this transfer is performed after the prior example, the transfer mode will still be `text`. Using the **ls** /+ command quickly confirms the active options.

❷ The **get** command uses the dataset path prefix `//` and pds member name in parentheses to identify the member to get. Note again that the dataset name for transfers is case insensitive.

Example: Get a PDS member

```
sftp> get //user.coz.sampjcl(cozproc) cozproc.txt          ❶
Fetching //user.coz.sampjcl(cozproc) to cozproc.txt

sftp> cd //user.coz.sampjcl     ❷
sftp> get runcoz     ❸
Fetching //USER.COZ.SAMPJCL/runcoz to runcoz

sftp> get *     ❹
Fetching //USER.COZ.SAMPJCL/@@README to @@README
Fetching //USER.COZ.SAMPJCL/BPXBATCH to BPXBATCH
Fetching //USER.COZ.SAMPJCL/BPXBATSL to BPXBATSL
Fetching //USER.COZ.SAMPJCL/COZCFGD to COZCFGD
Fetching //USER.COZ.SAMPJCL/COZPROC to COZPROC
Fetching //USER.COZ.SAMPJCL/DTLSPAWN to DTLSPAWN
Fetching //USER.COZ.SAMPJCL/GPGDSN to GPGDSN
Fetching //USER.COZ.SAMPJCL/GREPDSN to GREPDSN
Fetching //USER.COZ.SAMPJCL/GREPSED to GREPSED
Fetching //USER.COZ.SAMPJCL/OFFLDSMF to OFFLDSMF
Fetching //USER.COZ.SAMPJCL/RUNCOZ to RUNCOZ
Fetching //USER.COZ.SAMPJCL/RUNCOZ2 to RUNCOZ2
Fetching //USER.COZ.SAMPJCL/RUNCOZ3 to RUNCOZ3
Fetching //USER.COZ.SAMPJCL/RUNSPAWN to RUNSPAWN
Fetching //USER.COZ.SAMPJCL/RUNSPWN2 to RUNSPWN2
Fetching //USER.COZ.SAMPJCL/TDIRK to TDIRK
Fetching //USER.COZ.SAMPJCL/WGET2DSN to WGET2DSN
```

❶ The **get** can be used to get a member from a fully qualified dataset.
❷ The **cd** command is used to make a PDS the current working "directory".
❸ The **get** command uses just the member name to retrieve the desired member.
❹ The **get *** command can be used to retrieve all members at once.

Example: Put a text MVS dataset, overriding DCB attributes

```
...
sftp> ls /+mode=text,lrecl=80,recfm=fb     ❶
/+mode=text,lrecl=80,recfm=fb
sftp> put afile.txt //USER.AFILE.TXT     ❷
Uploading afile.txt to //USER.AFILE.TXT
```

❶ The option command `ls /+mode=text,lrecl=80,recfm=fb` is used to set the transfer mode to text, and set the DCB attributes for the new dataset `USER.AFILE.TXT`. This overrides the system default for new datasets. Input lines will be broken on CR, LF, or CRLF and lines longer than allowed by the dataset will be wrapped onto multiple records. The options `linerule` and `overflowrule` can be used to override those settings.
❷ The `put` command uses the specialized path prefix `//` (or, optionally `/-/`) to specify the dataset name.

Listing datasets and PDS directories

MVS datasets can be listed using the sftp **ls** command. Partitioned datasets are treated as directories with their members as entries.

In order to support existing sftp clients, several considerations have to be made when listing datasets:

- The **ls** lists multiple dataset levels (by default), and therefore can return a large amount of information. As such, listings that would involve searching the entire catalog, such as **ls //** or **ls //A*** are not allowed. Furthermore, because of the way sftp clients interact with the server, the following style of command is not supported: **ls //USER.LVL1***. However, the same effect can be produced by either using directory notation for searching the catalog (**ls //USER/LVL1***) or changing to the desired level and issuing a relative listing command:

```
cd //USER
ls LVL1*
```

- Where possible, dataset names are treated as case insensitive. A **get** or **put** can specify the name in either lower or upper case and it will be found. However, any globbed (wildcard) **ls** command requires upper case characters. Individual datasets can be listed in either upper or lower case. To be safe, it is a good idea to use upper case on all list requests.

- As mentioned above, partitioned datasets (PDS) are treated as directories. the long list form **ls -l** can be used to list member statistics, if statistics exist.

- The **ls** accepts dataset names prefixed either by `//` or `/-/`. The second form should be used for the few sftp clients that do not allow a double slash to sent to the server.

Example: Listing datasets

```
sftp> cd //USER
sftp> ls -al     ❶
Volume  Referred  Ext  Tracks    Used Recfm Lrecl BlkSz Dsorg  Dsname
WORK84 2008/09/05   1       1       1  FB      80 27920  PS    USER.AFILE.TXT
WORK81 2008/09/08   1      30       ?  U        0  6144  PO-E  USER.COZ.LOADLIB
WORK81 2008/09/11   1      15       4  FB      80 27920  PO    USER.COZ.SAMPJCL
WORK84 2008/09/11   1       1       1  U        0  6144  PS    USER.COZ.TEST.SEQ
WORK81 2008/09/09   1      15       3  FB      80 27920  PO    USER.COZ.TESTJCL

sftp> cd //USER     ❷
sftp> ls CO*
COZ.LOADLIB/          COZ.SAMPJCL/          COZ.TEST.SEQ          COZ.TESTJCL/

sftp> ls //USER/CO*     ❸
//USER/COZ.LOADLIB/   //USER/COZ.SAMPJCL/   //USER/COZ.TEST.SEQ
//USER/COZ.TESTJCL/

sftp> ls //USER.CO*     ❹
Couldn't get handle: Failure
Can't ls: "//USER.CO*" not found

sftp> ls //     ❺
Couldn't stat remote file: No such file or directory
Can't ls: "//" not found

sftp> cd //user
sftp> ls co*     ❻
Can't ls: "//USER/co*" not found
sftp>
```

❶ The long form of the list command **ls -al** will list detailed information from the catalog about each dataset.

❷ Relative listing requests can be performed by first navigating to the desired level, then issuing the list request without any prefix.

❸ When using wildcards, the desired result can be achieved by using a slash (`/` in place of the traditional level separator (`.`).

❹ Due to existing sftp client design, this list request would require the entire catalog to be searched, then filtered with the pattern `USER.CO*`. It is therefore disallowed.

❺ Lists that would involve the entire catalog are not supported. The `openssh` sftp client reports this as shown.

❻ The same command with a lower case pattern will fail (as described above).

Example: Listing a PDS directory

```
...
sftp> cd //user.coz.sampjcl     ❶
sftp> ls -al     ❷
Name           Size  Created         Changed          ID
@@README
BPXBATCH         13 2008/04/04 2008/04/04 17:18:09  USER
BPXBATSL         16 2008/04/03 2008/04/03 10:36:52  USER
COZCFGD          65 2008/03/27 2008/05/12 14:28:54  USER
COZPROC          30 2008/03/27 2008/03/27 11:54:48  USER
DTLSPAWN         40 2008/05/05 2008/05/05 09:31:08  USER
```

```
GPGDSN           15 2008/05/05 2008/05/05 10:40:05  USER
GREPDSN
GREPSED          12 2008/05/05 2008/05/05 09:30:51  USER
OFFLDSMF
RUNCOZ           20 2008/03/27 2008/05/12 14:08:02  USER
RUNCOZ2          15 2008/05/05 2008/05/05 10:02:51  USER
RUNCOZ3           8 2008/05/05 2008/05/06 08:50:37  USER
RUNSPAWN         54 2008/05/12 2008/05/12 14:25:37  USER
RUNSPWN2         20 2008/05/12 2008/05/12 13:19:05  USER
TDIRK            18 2008/04/03 2008/04/03 10:19:20  USER
WGET2DSN
```

❶ The **cd** command is used to make a PDS the current working "directory".

❷ The **ls -al** command (long list form) is used to display the members of the PDS, including available statistics.

3.4 Working with POSIX files

This section describes how to use the Co:Z implementation of sftp with POSIX files (HFS, zFS) on z/OS. Standard sftp implementations (including z/OS OpenSSH) support only binary mode file transfers. The Co:Z implementation provides binary transfer mode by default, but also supports text mode transfers. Text mode transfers are controlled via the following options:

- `mode`: when set to `text` causes file transfers to be text based.
- `clientcp` and `servercp`: When text mode is active, these settings determine the codepage translation that will take place. The default client code page is `ISO8859-1`. The default server code page is the current z/OS locale.
- `linerule`: When text mode is active, this setting determines how line separators are converted between the client and server.

Transferring Files

The `get` and `put` commands are used to transfer POSIX files (either on HFS or zFS filesystems).

The options (listed above) that have been previously set via the **ls /+option=value** are in effect for any given transfer. All other options (used for dataset support) are ignored for POSIX file transfers.

Example: Get a text POSIX file

```
sftp> ls /+mode=text,clientcp=UTF-8     ❶
/+mode=text,clientcp=UTF-8
sftp> ls /+     ❷
/+/clientcp=UTF-8     /+/mode=text          /+/servercp=IBM-1047
sftp> get .ssh/sftp-server.rc     ❸
Fetching /u/user/.ssh/sftp-server.rc to sftp-server.rc
/u/user/.ssh/sftp-server.rc                100%  234     0.2KB/s   00:00
sftp>
```

❶ The default transfer mode of binary is overridden and set to `text`. Additionally, the client code page is explicitly set to `UTF-8`.
❷ Displays the active options. Note that the server code page, if not explicitly set, defaults to the current z/OS locale.
❸ The **get** command requests the transfer of the POSIX file using the options in effect.

Example: Put a text POSIX file

```
sftp> put sftp-server.rc .ssh     ❶
Uploading sftp-server.rc to /home/user/.ssh/sftp-server.rc
sftp-server.rc                             100%  234     0.2KB/s   00:00
sftp>
```

❶ The client text file `sftp-server.rc` is put to the remote directory `.ssh` under the current working remote

directory. The active file transfer options are used.

3.5 Working with JES jobs and spool files

This section describes how to use Co:Z SFTP to submit jobs, query job status and access spool files on z/OS. Future releases of Co:Z SFTP will also support enhanced job cancel and purge facilities.

Note: Co:Z JES spool access supports both JES2 and JES3, but is currently limited to the primary JES subsystem.

Obtaining JES job status

To query the status of z/OS jobs, you simply list the "//-JES" pseudo-directory:

```
sftp> cd //-JES  ❶
sftp> ls  ❷
JOB00434  JOB00561  TSU00560  TSU00562
sftp> ls -al  ❸
JOBNAME  JOBID    OWNER    STATUS   CLASS
KIRKL    JOB00434 KIRK     OUTPUT   A        RC=0000
TOMCAT   JOB00561 KIRK     ACTIVE   A
KIRK     TSU00560 KIRK     OUTPUT   TSU      RC=0000
KIRK     TSU00562 KIRK     ACTIVE   TSU
sftp> ls /+jesjobname=kirk  ❹
/+jesjobname=kirk
sftp> ls -al
JOBNAME  JOBID    OWNER    STATUS   CLASS
KIRKL    JOB00434 KIRK     OUTPUT   A        RC=0000
KIRK     TSU00560 KIRK     OUTPUT   TSU      RC=0000
KIRK     TSU00562 KIRK     ACTIVE   TSU
sftp> ls /+jesjobname=kirk.  ❺
/+jesjobname=kirk.
sftp> ls -al
JOBNAME  JOBID    OWNER    STATUS   CLASS
KIRK     TSU00560 KIRK     OUTPUT   TSU      RC=0000
KIRK     TSU00562 KIRK     ACTIVE   TSU
sftp> ls /+nojesjobname  ❻
/+nojesjobname
sftp> ls /+jesowner=goetze  ❼
/+jesowner=goetze
sftp> ls -al
JOBNAME  JOBID    OWNER    STATUS   CLASS
GOETZEB  JOB00601 GOETZE   OUTPUT   A        RC=0000
GOETZE   TSU00505 GOETZE   OUTPUT   TSU      RC=0000
GOETZE   TSU00515 GOETZE   ACTIVE   TSU
sftp> ls /+jesstatus=active  ❽
/+jesowner=goetze
sftp> ls -al
JOBNAME  JOBID    OWNER    STATUS   CLASS
GOETZE   TSU00515 GOETZE   ACTIVE   TSU
```

❶ Change to the `//-JES` pseudo-directory.

❷ Listing the contents of the `//-JES` directory will by default display a list a job ids whose owner is the same as the current user.

❸ Requesting a detailed listing of the `//-JES` directory produces a formatted list of the same jobs. Note here how jobs are sorted lexically by jobid - this is actually being done by the sftp client. Sorting on most sftp clients can be disabled; in the case of OpenSSH, use the `-f` switch on the `ls` command, eg: `ls -alf` will display the jobs in the order returned by the JES subsystem interface.

❹ By default, all jobs owned by the current user are displayed. The `jesjobname` setting may be used to set a jobname filter.

❺ Terminating the `jesjobname` setting with a period filters on an exact jobname match, rather than a prefix.

❻ The `jesjobname` setting is turned off.
❼ By default, the `jesowner` setting is set to the current userid. Here it is changed to a different userid.
❽ The `jesstatus` setting may be used to filter job listings by one of the following categories: `input`, `output`, or `active`.

Co:Z SFTP uses the unauthorized "Extended Status" subsystem interface to obtain job status. This facility is only available if you are running z/OS 1.9 or later. A SAF(RACF) SECLABEL dominance check may be used by the IBM extended status subsystem interface to control access to this facility; refer to RACF or your security product documentation for more information.

Transferring JES spool files

Job spool files may be transferred using normal SFTP "get" commands from your SFTP client.

```
sftp> cd //-JES
sftp> ls -al
JOBNAME  JOBID    OWNER    STATUS   CLASS
KIRKL    JOB00434 KIRK     OUTPUT   A        RC=0000
TOMCAT   JOB00561 KIRK     ACTIVE   A
KIRK     TSU00560 KIRK     OUTPUT   TSU      RC=0000
KIRK     TSU00562 KIRK     ACTIVE   TSU
sftp> cd JOB00434  ❶
sftp> ls  ❷
102  2    3    4
sftp> ls -al  ❸
DSID STEPNAME PROCSTEP DDNAME   C OWNER    RECFM LRECL BYTES
 102 LOGDEF            SYSPRINT A KIRK     FBA     133 5195
   2 JES2              JESMSGLG A KIRK     FA      133 911
   3 JES2              JESJCL   A KIRK     V       136 271
   4 JES2              JESYSMSG A KIRK     VA      137 839
sftp> ls -alf  ❹
DSID STEPNAME PROCSTEP DDNAME   C OWNER    RECFM LRECL BYTES
   2 JES2              JESMSGLG A KIRK     FA      133 911
   3 JES2              JESJCL   A KIRK     V       136 271
   4 JES2              JESYSMSG A KIRK     VA      137 839
 102 LOGDEF            SYSPRINT A KIRK     FBA     133 5195
sftp> ls /+mode=text  ❺
/+mode=text
sftp> get 102 logdef.text  ❻
Fetching //-JES.JOB00434/102 to logdef.text
//-JES.JOB00434/102
sftp>
sftp> get *  ❼
Fetching //-JES.JOB00434/102 to 102
//-JES.JOB00434/102
Fetching //-JES.JOB00434/2 to 2
//-JES.JOB00434/2
Fetching //-JES.JOB00434/3 to 3
//-JES.JOB00434/3
Fetching //-JES.JOB00434/4 to 4
//-JES.JOB00434/4
sftp>
sftp> get all concat.txt ❽
Fetching //-JES.JOB00434/all to concat.txt
sftp> get jesysmsg  ❾
Fetching //-JES.JOB00434/jesysmsg to jesysmsg
sftp> get logdef.sysprint
Fetching //-JES.JOB00434/logdef.sysprint to logdef.sysprint
```

❶ Jobs are represented in Co:Z SFTP as *directories* under the `//-JES` pseudo-directory. Here we change the current directory to a specific job.
❷ Job spool files are represented as file names with the numeric JES `DSID` identifier.
❸ A detailed listing displays a formatted list of spool files.
❹ Many sftp clients will sort the files lexically by name (dsid). The `ls -f` switch on the OpenSSH sftp client will preserve the natural ordering, which is by numerical dsid.
❺ The transfer mode is set to `text`.
❻ The sftp **get** command can be used to download spool files.
❼ A wildcard get command can be used to download all spool files in the job directory.
❽ The special `ALL` file name can be used to transfer all spool files to a concatenated output file. To ensure that the correct transfer options are set when using the `ALL` file name, it is recommended that the following pattern be defined in the system `cozsftp_server_config` configuration file:

```
pattern: //-JES.*.ALL
mode=text,jesincsysin
```

❾ Spool files may also be referenced by `[step.[procstep.]]ddname`

Using the `jesincsysin` option (available on z/OS 1.10 or later):

```
sftp> ls /+jesincsysin  ❶
/+jesincsysin
sftp> ls -alf
DSID STEPNAME PROCSTEP DDNAME   C OWNER    RECFM LRECL BYTES
   1                   JESJCLIN A KIRK     F        80 316
   2 JES2              JESMSGLG A KIRK     FA      133 911
   3 JES2              JESJCL   A KIRK     V       136 271
   4 JES2              JESYSMSG A KIRK     VA      137 839
 101 LOGDEF            SYSIN    A KIRK     F        80 177
 102 LOGDEF            SYSPRINT A KIRK     FBA     133 5195

sftp> get all concat.txt ❷
Fetching //-JES.JOB00434/all to concat.txt
sftp>sftp> get jesjclin ❸
Fetching //-JES.JOB00434/JESJCLIN to jesjclin
```

❶ The `jesincsysin` option may be used to specify that the spool file listings and concatenated output will contain SYSIN spool files, including JESJCLIN.
❷ When option `jesincsysin` is enabled, concatenated spool file downloads will include SYSIN spool files. In addition, the separator between spool files will be annotated with the [step.[procstep.]]ddname. This option is a convenient way to download all job input and output.
❸ Regardless of how option `jesincsysin` is set, you can download individual SYSIN spool files. When downloading JESJCLIN as an individual spool file, the output will contain the other SYSIN spool files embedded in the orignal JCL.

Co:Z SFTP uses the unauthorized interface to the JES "Spool Browse" facility, which is only available if you are running z/OS 1.9 or later. As with IBM FTP, the SAF(RACF) JESSPOOL resource class is used to control access to spool files throuh the Spool Browse facility.

Submitting JES jobs

Jobs may be submitted to the JES internal reading using SFTP "put" commands from your SFTP client into a special pseudo-directory named "//-JES.INTRDR".

```
sftp> ls /+mode=text
/+mode=text
sftp> !cat jcl.txt  ❶
//SLEEP3  JOB (),'Kirk Wolf',MSGCLASS=H
//UNIX    EXEC PGM=COZBATCH
//STEPLIB DD DISP=SHR,DSN=KIRK.COZ.LOADLIB
//STDIN   DD *
for i in 1 2 3
do
  echo "Sleeping..."
  sleep 1
done
//
sftp> cd //-jes.intrdr  ❷
sftp> put jcl.txt myjob  ❸
Uploading jcl.txt to //-JES.INTRDR/myjob

sftp> ls -al  ❹
ALIAS    JOBNAME  JOBID    OWNER    STATUS   CLASS    COMPL
MYJOB    SLEEP3   JOB01941 KIRK     ACTIVE   A

sftp> ls -al
ALIAS    JOBNAME  JOBID    OWNER    STATUS   CLASS    COMPL
MYJOB    SLEEP3   JOB01941 KIRK     OUTPUT   A        RC=0000

sftp> cd myjob  ❺
sftp> ls -alf
DSID STEPNAME PROCSTEP DDNAME   C OWNER    RECFM LRECL BYTES
   2 JES2              JESMSGLG H KIRK     FA      133 1316
   3 JES2              JESJCL   H KIRK     V       136 373
   4 JES2              JESYSMSG H KIRK     VA      137 824
 102 UNIX              SYSOUT   H KIRK     FBA     121 308

sftp> get 2  2.txt
Fetching //-JES.INTRDR.MYJOB/2 to /tmp/2.txt

sftp> get all jobout.txt
Fetching //-JES.INTRDR.MYJOB/all to /tmp/jobout.txt

sftp> cd ..
sftp> put jcl.txt myjob2
sftp> ls -al
ALIAS    JOBNAME  JOBID    OWNER    STATUS   CLASS    COMPL
MYJOB    SLEEP3   JOB01941 KIRK     OUTPUT   A        RC=0000
MYJOB2   SLEEP3   JOB01943 KIRK     ACTIVE   A

sftp> ls /+jesjobwait  ❻
```

```
/+jesjobwait
sftp> cd myjob2
sftp> get all jobout2.txt
Fetching //-JES.INTRDR.MYJOB2/all to /tmp/jobout2.txt

sftp> ls /+jesjobwait=10.1  ❼
sftp> ls /+nojesjobwait
```

❶ Run the `cat` command on the sftp client to display a file containing JCL.
❷ `//-JES.INTRDR` is a special pseudo-directory that contains any jobs submitted by the current session.
❸ The JCL is submitted by uploading it using the sftp client's `put` command. The target file name `MYJOB` is a *handle* that can be used to refer to this job later in the same session.
❹ Listing the `//-JES.INTRDR` directory displays all of the jobs that have been submitted in this session.
❺ `//-JES.INTRDR.MYJOB` is a directory that contains all of the spool files for the job referenced by this handle.
❻ The `jesjobwait` setting can be used to cause Co:Z SFTP server to wait until the job completes before listing or transferring the jobs spool files.
❼ The default time limit to wait is 60 seconds with a polling interval of 2 seconds (60.2), but this can also be changed.

JES related options

The following table describes options that affect JES submit, status, and spool file transfer.

Table 3.1. JES related options

Name	Value	Notes
jesjobname	<pattern>	The value of this setting is used as a filter when listing jobs. If the value doesn't end in a period, then the value is used as the job prefix The default for this setting is `nojesjobname`, which means that jobs are not filtered by name.
jesowner	<userid>	This setting specifies the userid used to filter job listings by job owner. The default for this setting is the current MVS userid, but may be set to `nojesowner` or `jesowner=*` to disable filtering by owner.
jesstatus	input|output|active	This setting is used to filter job listings by job status. The default for this setting is `nojesstatus`.
jesjobwait	secs[.intvl]	If enabled, this setting specifies the time in seconds to wait for a job to complete before listing or transferring its spool files. For most sftp clients, a `cd` to the job's spool file directory will also wait. If no value is specified, the default is `60.2`, which means to wait up to 60 seconds, polling every 2 seconds. The default for this setting is `nojesjobwait`.
jeslrecl	<numeric>	This setting specifies the lrecl used when submitting jobs to the

Name	Value	Notes
		internal reader. The default for this setting is `80`.
jesrecfm	f|fb|v|vb	This setting specifies the record format used when submitting jobs to the JES internal reader. The default for this setting is `F`.
jesincsysin		When enabled, listings and concatenated downloads of JES spool files will include SYSIN spool files. Available on z/OS 1.10 or later. The default for this setting is `NOjesincsysin`.

4. Using the Co:Z SFTP client

An enhanced sftp client (**cozsftp**) for z/OS is also included in the Co:Z toolkit. This client can be used to initiate transfers with a remote host and supports the same set of file transfer options as the Co:Z SFTP server. The **cozsftp** command is installed in the **$COZ_HOME/bin** directory.

4.1 Starting the Co:Z SFTP client on z/OS

```
$ export PATH=/opt/dovetail/coz/bin:$PATH ❶
$ cozsftp user@host
Co:Z sftp version: 1.1.0 (5.0p1) 2008-10-20
Copyright (C) Dovetailed Technologies, LLC. 2008. All rights reserved.
Connecting to host...
user@host's password: *****
cozsftp>
```

❶ Add the Co:Z binaries directory to your PATH. This is not necessary if symbolic links from **/bin** were created during installation.

4.2 Co:Z SFTP client logging

OpenSSH SFTP logging is enabled by setting the `-v` option on the command line. The logging level can be raised by increasing the number of `v` options from `-v` to `-vvv`. Additionally, the `COZ_LOG` environment variable can be used to set logging options for the Co:Z extension library used to add z/OS support to the **cozsftp** command. The default setting for `COZ_LOG` is `I`. To change this default for all users, modify /etc/ssh/cozsftp_client.rc as needed. Individual users can override this setting by exporting COZ_LOG on the command line or in the copy of `cozsftp_client.rc` in their individual .ssh directory.

Client logging information is sent to `stderr`. In order to capture logging information in a file, use the following:

```
$ export COZ_LOG=D; cozsftp -vvv user@hostname  2>&1  | tee ~/cozsftp-output.log
```

The Co:Z client `loglevel` can also be set using the **lzopts** command described in the next section. See *Section B.2, "Miscellaneous options"* for additional information.

For information on setting these logging options in batch, see *the section called "Logging in batch"*.

4.3 Setting, displaying and clearing file transfer options

The enhanced client introduces two new commands:

`lzopts` [-a] [*`option=value,...`*]

The **lzopts** command is used to set local (client) file transfer options. These options are set prior to initiating file/dataset transfers from z/OS to a remote host.

`zopts` [-a] [*`option=value,...`*]

The **zopts** command is used to set server file transfer options -- if the server is a Co:Z SFTP server. The zopts command is functionally equivalent to the **ls /+<option_list>** command used by existing clients to set Co:Z sftp-server file transfer options.

Multiple options can be set by separating the option=value pairs with commas. An error is returned if one or more of the options was incorrectly specified, but the remaining options are set as requested.

The active options and their settings can be displayed by issuing the commands without arguments. The `-a` option can be specified to list *all* available options, even those that are not active.

The client performs some shell-like processing of its commands. In general, this is not an issue for the setting of options, but if the supplied option value contains a hash symbol (#), the option=value pair must be quoted, either with single or double quotes. For example:

```
cozsftp> lzopts "dataclas=#MYCLASS"
```

Note

For compatibility with z/OS OpenSSH SFTP, the **cozsftp** command recognizes the following additional subcommands: **ascii** and **binary**. these subcommands are treated as synonyms for `lzopts mode=text` and `lzopts mode=binary` respectively.

Client session options are determined in the following priority order:

1. The `fixed:` section of `/etc/ssh/cozsftp_config` (highest priority and non-modifiable)
2. The first matching pattern (if any) from `$HOME/.ssh/cozsftp_config`
3. The first matching pattern (if any) from `/etc/ssh/cozsftp_config`
4. Previous interactive commands: `lzopts` (described below) in the same session
5. The environment variable `SFTP_ZOS_OPTIONS`
6. The `default:` section of `/etc/ssh/cozsftp_config` (lowest priority)

For a list of available options, see *Appendix B, Co:Z SFTP options*.

For a description of the `cozsftp_config` file format, including how to specify file name patterns, see *Appendix C, Session config files*.

Example: Setting and displaying local (client) transfer options

```
cozsftp> lzopts mode=text     ❶
 mode=text
cozsftp> lzopts     ❷
clientcp=IBM-1047    loglevel=I           mode=text
servercp=IBM-1047
```

❶ The local option command `lzopts mode=text` is used to set the transfer mode to text. **mode=binary** is the default.

❷ The local option list command `lzopts` shows the options currently in effect. In this case, the codepages `clientcp` and `servercp` are set to the defaults.

Example: Setting multiple local options

```
cozsftp> lzopts lrecl=80,recfm=fb,space=trk.3.2     ❶
 lrecl=80,recfm=fb,space=trk.3.2
```

❶ Multiple options can be specified, separated by commas. Note that the SPACE parameter uses periods for commas to avoid ambiguity.

Example: Showing all local options

```
cozsftp> lzopts -a      ❶
clientcp=IBM-1047  linerule=flexible  loglevel=I        lrecl=80
mode=text          overflow=wrap      recfm=fb          servercp=IBM-1047
space=trk.3.2      NOallowmount       NOblksize         NObufno
NOcopies           NOdataclas         NOdest            NOdir
NOdisp             NOdsorg            NOforms           NOgdgnt
NOhold             NOlabel            NOlike            NOmaxvol
NOmgmtclas         NOnorecall         NOoutdes          NOrelease
NOretpd            NOsequence         NOshowall         NOspin
NOstorclas         NOsysout           NOtrim            NOtrtch
NOucount           NOunit             NOvol             NOwriter
```

❶ The option command **lzopts -a** is used to show all of the available options, even those that are not currently active.

4.4 Coordinating Transfer Options with a Co:Z SFTP Server

The enhanced Co:Z SFTP client can connect to any sftp server, including a Co:Z SFTP server. In this case, there are two sets of transfer options in effect; the enhanced client's and the server's. Client side (local) options are controlled via the **lzopts** command. Server side (remote) options are controlled via the **zopts** command.

When transferring POSIX files between a z/OS server and z/OS client, using the default `mode=binary` transfer option both locally and remotely will usually yield the desired results. If codepage translations need to take place, the desired `clientcp`, `servercp` and `mode=text` can be set either locally (via **lzopts** command) or remotely (via the **zopts** command). The other side can be left in `mode=binary`.

When transferring datasets between a z/OS server and z/OS client, it is generally recommended that `linerule=rdw` be used for binary transfers so that record mode boundaries are preserved.

When converting from dataset to POSIX file between a z/OS server and z/OS client, the transfer options should be set where the dataset resides.

4.5 Working with Datasets

The Co:Z implementation of sftp accepts two prefix strings to identify MVS datasets as absolute paths. The first (`//`) is consistent with IBM's common usage. A secondary form (`/-/`) is also available.

Navigating Datasets

The sftp **lcd** command can be used to navigate around the z/OS dataset space. Using the dataset prefix `//` or `/-/`, the dataset space can be entered. Once there, traversal up and down various dataset levels can be performed similarly to hierarchical file systems.

Partitioned datasets are treated as directories as well. Once a PDS is made the current working directory, its members can be listed and retrieved like normal files.

Just as listing the entire catalog from the root is not allowed, it is not possible to make the catalog root the current working directory. As such, the command **lcd //** will fail.

Example: Navigating the dataset space

```
cozsftp> lcd //user     ❶
cozsftp> lpwd     ❷
Local working directory: //USER
cozsftp> lcd coz.testjcl     ❸
cozsftp> lpwd
Local working directory: //USER.COZ.TESTJCL
cozsftp> lcd ..     ❹
cozsftp> lpwd
Local working directory: //USER.COZ
```

❶ Using the dataset prefix `//`, the high level qualifier `user` is specified. For lcd commands, the dataset name is case insensitive.

❷ The **lpwd** command will list the current working dataset level. Note that the name is properly displayed in uppercase

❸ Multiple levels can be traversed at a time. Instead of using the normal separator (`.`), a slash can be used: **lcd coz/testjcl**.

❹ The **lcd ..** command will move up a level, as expected.

Transferring Datasets

The `get` and `put` commands are used to transfer datasets and PDS members.

Any options previously set via the **lzopts** are in effect for any given transfer.

Example: Get a file to a text sequential dataset

```
$ cozsftp user@linux.com     ❶
Connecting to linux.com...
user@linux.com's password:
cozsftp> lzopts mode=text     ❷
 mode=text
cozsftp> lzopts
clientcp=IBM-1047    loglevel=I           mode=text
servercp=ISO8859-1
cozsftp> get /tmp/GPGDSN //USER.GPGDSN     ❸
Fetching /tmp/GPGDSN to //USER.GPGDSN
ZosDataset[I]: Opening dataset USER.GPGDSN for write with options: new catalog
/tmp/GPGDSN                              100% 1215     1.2KB/s   00:00
ZosDataset[I]: Closing dataset //USER.GPGDSN - 1215 bytes received, 15 records written
ZosSmf119Record[I]: SMF Type119 recording not enabled; SMF recording disabled
```

❶ This example shows the full connection process, using keyboard-interactive password authentication to a remote linux system.

❷ The default transfer mode of binary is overridden and set to `text`.

❸ The `get` command uses the dataset path prefix `//` (or, optionally `/-/`) to specify that a dataset is to be written. At the default log level of `I` (INFO), information is emitted about the transfer process. Note also that in this case, SMF recording is disabled because the FTP SMF records (type 119) are not currently configured for recording.

Example: Get a text file to a PDS member

```
cozsftp> lzopts     ❶
clientcp=IBM-1047     loglevel=I     mode=text
servercp=ISO8859-1
cozsftp> lcd //user.coz.testjcl
cozsftp> lpwd
Local working directory: //USER.COZ.TESTJCL
cozsftp> get /tmp/GPGDSN     ❷
Fetching /tmp/GPGDSN to //USER.COZ.TESTJCL/GPGDSN
ZosDataset[I]: Opening dataset USER.COZ.TESTJCL(GPGDSN) for write with options: old
/tmp/GPGDSN                                          100% 1215     1.2KB/s   00:00
ZosDataset[I]: Closing dataset //USER.COZ.TESTJCL(GPGDSN) - 1215 bytes received, 15 records written
```

❶ If this transfer is performed after the prior example, the transfer mode will still be `text`. Using the **lzopts** command quickly confirms the active options.

❷ The **get** command uses the dataset path prefix `//` and pds member name in parentheses to identify the member to create.

Example: Get multiple files using a wild-card pattern to a GDG

In *release 2.4.0*, support was added to allow multiple files to be downloaded to new generations of a GDG.

```
cozsftp> lzopts gdgnt     ❶
gdgnt
cozsftp> get /tmp/*.data //USER.COZ.GDG(+1)     ❷
Fetching /tmp/file1.data to //USER.COZ.GDG(+1)
ZosDataset[I]: Opening dataset //USER.COZ.GDG(+1) for write with options: new catalog
/tmp/file1.data                                          100% 523     20.2KB/s   00:01
ZosDataset[I]: Closing dataset //USER.COZ.GDG.G0001V00 - 523 bytes received, 10 records written ❸
Fetching /tmp/test2.data to //USER.COZ.GDG(+1)
ZosDataset[I]: Opening dataset //USER.COZ.GDG(+1) for write with options: new catalog
/tmp/test2.data                                          100% 886     18.5KB/s   00:01
ZosDataset[I]: Closing dataset //USER.COZ.GDG.G0002V00 - 886 bytes received, 12 records written
```

❶ Wild-card downloading of remote files to new GDG generations is only supported if the `gdgnt` option is enabled. Sites should consider adding this option to the `default` section of their `/etc/ssh/cozsftp_server_config` file.

❷ This **get** command uses a wild-card (*) pattern to select any file in the `/tmp` directory that ends in `".data"`. Each file will be downloaded to a new generation of the target GDG: `USER.COZ.GDG`.

❸ Each file that matches the pattern is transferred separately. The generation name that was used is printed when the data set is closed.

Example: Put PDS members

```
cozsftp> lpwd
Local working directory: //USER.COZ.TESTJCL
cozsftp> put ONETEST /tmp/ONETEST     ❶
Uploading ////USER.COZ.TESTJCL(ONETEST) to /tmp/ONETEST
ZosDataset[I]: Opening dataset USER.COZ.TESTJCL(ONETEST) for read with options: shr

ZosDataset[I]: Closing dataset //USER.COZ.TESTJCL(ONETEST) - 38 records read, 3078 bytes sent

cozsftp> put //USER.coz.testjcl(*)     ❷
Uploading //USER.COZ.TESTJCL(@@README) to /tmp/@@README
ZosDataset[I]: Opening dataset USER.COZ.TESTJCL(@@README) for read with options: shr

ZosDataset[I]: Closing dataset //USER.COZ.TESTJCL(@@README) - 34 records read, 2754 bytes sent
Uploading //USER.COZ.TESTJCL(ALLOCDS) to /tmp/ALLOCDS
ZosDataset[I]: Opening dataset USER.COZ.TESTJCL(ALLOCDS) for read with options: shr

ZosDataset[I]: Closing dataset //USER.COZ.TESTJCL(ALLOCDS) - 6 records read, 486 bytes sent
Uploading //USER.COZ.TESTJCL(CHKENVD) to /tmp/CHKENVD
ZosDataset[I]: Opening dataset USER.COZ.TESTJCL(CHKENVD) for read with options: shr

ZosDataset[I]: Closing dataset //USER.COZ.TESTJCL(CHKENVD) - 1 records read, 81 bytes sent
Uploading //USER.COZ.TESTJCL(CHKPOST) to /tmp/CHKPOST
ZosDataset[I]: Opening dataset USER.COZ.TESTJCL(CHKPOST) for read with options: shr

ZosDataset[I]: Closing dataset //USER.COZ.TESTJCL(CHKPOST) - 6 records read, 486 bytes sent
Uploading //USER.COZ.TESTJCL(CHKPRE) to /tmp/CHKPRE
ZosDataset[I]: Opening dataset USER.COZ.TESTJCL(CHKPRE) for read with options: shr

ZosDataset[I]: Closing dataset //USER.COZ.TESTJCL(CHKPRE) - 72 records read, 5832 bytes sent
Uploading //USER.COZ.TESTJCL(COZCFGO) to /tmp/COZCFGO
ZosDataset[I]: Opening dataset USER.COZ.TESTJCL(COZCFGO) for read with options: shr

ZosDataset[I]: Closing dataset //USER.COZ.TESTJCL(COZCFGO) - 1 records read, 81 bytes sent
Uploading //USER.COZ.TESTJCL(GPGDSN) to /tmp/GPGDSN
ZosDataset[I]: Opening dataset USER.COZ.TESTJCL(GPGDSN) for read with options: shr
```

```
ZosDataset[I]: Closing dataset //USER.COZ.TESTJCL(GPGDSN) - 15 records read, 1215 bytes sent
Uploading //USER.COZ.TESTJCL(ONETEST) to /tmp/ONETEST
ZosDataset[I]: Opening dataset USER.COZ.TESTJCL(ONETEST) for read with options: shr

ZosDataset[I]: Closing dataset //USER.COZ.TESTJCL(ONETEST) - 38 records read, 3078 bytes sent
Uploading //USER.COZ.TESTJCL(TESTPROC) to /tmp/TESTPROC
ZosDataset[I]: Opening dataset USER.COZ.TESTJCL(TESTPROC) for read with options: shr

ZosDataset[I]: Closing dataset //USER.COZ.TESTJCL(TESTPROC) - 111 records read, 8991 bytes sent
Uploading //USER.COZ.TESTJCL(USERTEST) to /tmp/USERTEST
ZosDataset[I]: Opening dataset USER.COZ.TESTJCL(USERTEST) for read with options: shr

ZosDataset[I]: Closing dataset //USER.COZ.TESTJCL(USERTEST) - 187 records read, 15147 bytes sent
```

❶ In this case, the current local directory is the PDS. This **put** command will transfer a specific member from a fully qualified dataset. Alternatively, the command: **put //USER.COZ.TESTJCL(ONETEST) /target** could be used without regard to the current local directory.

❷ When the **put** command is used on a PDS with "*" specified as the member, all of the members are uploaded. Note that the ability to specify a mask, like `(AB*)` is not currently supported.

Example: Put all generations of a GDG

In *release 2.4.0*, support was added to allow all generations of a GDG to be uploaded in one put commmand.

```
cozsftp> ls -alf //coz.test.gdg
Volume   Referred  Ext  Tracks    Used Recfm Lrecl BlkSz Dsorg  Dsname
                                                     GDG   COZ.TEST.GDG
VPWRKA  2013/06/04   1       1       1  U        0  6144  PS    COZ.TEST.GDG.G0003V00
VPWRKA  2013/06/04   1       1       1  U        0  6144  PS    COZ.TEST.GDG.G0004V00
VPWRKC  2013/06/04   1       1       1  U        0  6144  PS    COZ.TEST.GDG.G0005V00
VPWRKB  2013/06/04   1       1       1  U        0  6144  PS    COZ.TEST.GDG.G0006V00

cozsftp> put //coz.test.gdg(*) /tmp  ❶
Uploading //COZ.TEST.GDG.G0003V00 to /tmp/G0003V00
ZosDataset[I]: Opening dataset COZ.TEST.GDG.G0003V00 for read with options: shr
//COZ.TEST.GDG.G0003V00                                    20%   10KB  10.0KB/s   00:03 ETA
ZosDataset[I]: Closing dataset //COZ.TEST.GDG.G0003V00 - 2 records read, 10248 bytes sent

Uploading //COZ.TEST.GDG.G0004V00 to /tmp/G0004V00
ZosDataset[I]: Opening dataset COZ.TEST.GDG.G0004V00 for read with options: shr
//COZ.TEST.GDG.G0004V00                                    20%   10KB  10.0KB/s   00:03 ETA
ZosDataset[I]: Closing dataset //COZ.TEST.GDG.G0004V00 - 2 records read, 10248 bytes sent

Uploading //COZ.TEST.GDG.G0005V00 to /tmp/G0005V00
ZosDataset[I]: Opening dataset COZ.TEST.GDG.G0005V00 for read with options: shr
//COZ.TEST.GDG.G0005V00                                     0%    5     0.0KB/s 2:43:49 ETA
ZosDataset[I]: Closing dataset //COZ.TEST.GDG.G0005V00 - 1 records read, 5 bytes sent

Uploading //COZ.TEST.GDG.G0006V00 to /tmp/G0006V00
ZosDataset[I]: Opening dataset COZ.TEST.GDG.G0006V00 for read with options: shr
//COZ.TEST.GDG.G0006V00                                     0%    5     0.0KB/s 2:43:49 ETA
ZosDataset[I]: Closing dataset //COZ.TEST.GDG.G0006V00 - 1 records read, 5 bytes sent
```

❶ In this example, all generations of a GDG are uploaded to the `/tmp` directory on the target system. As with all wild-card put commands, if the target directory is not specified it defaults to the current remote working directory.

Listing datasets and PDS directories

MVS datasets can be listed using the sftp **lls** command. Partitioned datasets are treated as directories with their members as entries.

When listing z/OS datasets locally with the **lls** command, catalog search filter keys are in effect for any wildcard requests. The catalog search wildcards `*`, `**`, and `%` used in the examples below are described in the IBM manual *DFSMS: Managing Catalogs - SC26-7409*. Note that this is different behavior from sftp clients that connect to the Co:Z sftp-server and list datasets with the **ls**. In that case, regular file globbing rules are in effect.

Example: Listing datasets

```
cozsftp> lcd //USER
cozsftp> lls -al     ❶
Volume  Referred  Ext  Tracks    Used Recfm Lrecl BlkSz Dsorg  Dsname
WORK84 2008/09/05   1       1       1  FB      80 27920  PS    USER.AFILE.TXT
WORK81 2008/09/08   1      30       ?  U        0  6144  PO-E  USER.COZ.LOADLIB
WORK81 2008/09/11   1      15       4  FB      80 27920  PO    USER.COZ.SAMPJCL
WORK84 2008/09/11   1       1       1  U        0  6144  PS    USER.COZ.TEST.SEQ
WORK81 2008/09/09   1      15       3  FB      80 27920  PO    USER.COZ.TESTJCL

cozsftp> lls -al //user.coz.t*     ❷
Volume  Referred  Ext  Tracks    Used Recfm Lrecl BlkSz Dsorg  Dsname
WORK81 2008/10/20   1      15       4  FB      80 27920  PO    USER.COZ.TESTJCL

cozsftp> lls -al //user.c*.**     ❸
Volume  Referred  Ext  Tracks    Used Recfm Lrecl BlkSz Dsorg  Dsname
WORK81 2008/10/20   1      30       ?  U        0  6144  PO-E  USER.COZ.LOADLIB
WORK81 2008/10/20   1      15       4  FB      80 27920  PO    USER.COZ.SAMPJCL
WORK84 2008/09/25   1       1       1  U        0  6144  PS    USER.COZ.TEST.SEQ
WORK81 2008/10/20   1      15       4  FB      80 27920  PO    USER.COZ.TESTJCL
```

❶ The long form of the list command **ls -al** will list detailed information from the catalog about each dataset.

❷ Using the catalog search filter key syntax, a single asterisk can be used to as a wildcard for a single dataset level.

❸ Using the catalog search filter key syntax, a double asterisk can be used to perform a *deep* listing. In this example, the single and double asterisk syntax is combined to list all of the datasets beginning with the prefix `USER.C`.

Example: Listing a PDS directory

```
...
cozsftp> lcd //user.coz.sampjcl     ❶
cozsftp> lls -al     ❷
Name           Size  Created         Changed          ID
@@README
BPXBATCH         13 2008/04/04 2008/04/04 17:18:09  USER
BPXBATSL         16 2008/04/03 2008/04/03 10:36:52  USER
COZCFGD          65 2008/03/27 2008/05/12 14:28:54  USER
COZPROC          30 2008/03/27 2008/03/27 11:54:48  USER
DTLSPAWN         40 2008/05/05 2008/05/05 09:31:08  USER
GPGDSN           15 2008/05/05 2008/05/05 10:40:05  USER
```

```
GREPDSN
GREPSED          12 2008/05/05 2008/05/05 09:30:51  USER
OFFLDSMF
RUNCOZ           20 2008/03/27 2008/05/12 14:08:02  USER
RUNCOZ2          15 2008/05/05 2008/05/05 10:02:51  USER
RUNCOZ3           8 2008/05/05 2008/05/06 08:50:37  USER
RUNSPAWN         54 2008/05/12 2008/05/12 14:25:37  USER
RUNSPWN2         20 2008/05/12 2008/05/12 13:19:05  USER
TDIRK            18 2008/04/03 2008/04/03 10:19:20  USER
WGET2DSN
```

❶ The **lcd** command is used to make a PDS the current working local "directory".
❷ The **lls -al** command (long list form) is used to display the members of the PDS, including available statistics.

4.6 Working with POSIX files

This section describes how to use the enhanced client with POSIX files (HFS, zFS) on z/OS. Standard sftp implementations (including z/OS OpenSSH) support only binary mode file transfers. The Co:Z implementation provides binary transfer mode by default, but also supports text mode transfers. Text mode transfers are controlled via the following options:

- `mode`: when set to `text` causes file transfers to be text based.
- `clientcp` and `servercp`: When text mode is active, these settings determine the codepage translation that will take place. The default client code page is `ISO8859-1`. The default server code page is the current z/OS locale.
- `linerule`: When text mode is active, this setting determines how line separators are converted between the client and server.

Transferring Files

The `get` and `put` commands are used to transfer POSIX files (either on HFS or zFS filesystems).

The options (listed above) that have been previously set via the **lzopts** are in effect for any given transfer. All other options (used for dataset support) are ignored for POSIX file transfers.

Example: Get a text POSIX file

```
$ cozsftp user@linux.com     ❶
Connecting to linux.com...
user@linux.com's password: *****
cozsftp> lzopts mode=text,servercp=UTF-8     ❷
 mode=text         servercp=UTF-8
cozsftp> lzopts     ❸
clientcp=IBM-1047    loglevel=I           mode=text
servercp=UTF-8
cozsftp> pwd
Remote working directory: /tmp
cozsftp> get msgs.txt     ❹
Fetching /tmp/msgs.txt to msgs.txt
/tmp/msgs.txt                                         100%   19KB  19.0KB/s   00:00
ZosPosixFile[I]: Closing file msgs.txt - 19488 bytes received, 19488 bytes written❺
cozsftp>
```

❶ This example shows the full connection process, using keyboard-interactive password authentication to a remote linux system.

❷ The default transfer mode of binary is overridden and set to `text`. Additionally, the server (linux) code page is explicitly set to `UTF-8`.

❸ Displays the active options. Note that the client code page, if not explicitly set, defaults to the current z/OS locale.

❹ The **get** command requests the transfer of the POSIX file using the options in effect.

❺ Upon completion, an informational message is written that describes the number of bytes received from the server and the number of bytes written to the local file. These counts are commonly the same, but changes in line separators and codepages can result in different counts.

Example: Put a text POSIX file

```
cozsftp> put sftp-server.log /tmp     ❶
Uploading sftp-server.log to /tmp/sftp-server.log
sftp-server.log                                   100%  127     0.1KB/s   00:00
ZosPosixFile[I]: Closing file sftp-server.log - 127 bytes read, 127 bytes sent
ZosSmf119Record[I]: SMF Type119 recording not enabled; SMF recording disabled
```

❶ The client text file `sftp-server.log` is put to the remote directory `/tmp`. The active file transfer options are used.

4.7 Using the Co:Z SFTP client in batch

The **cozsftp** client command can be conveniently used in a z/OS batch job without user interaction. The **COZBATCH** batch utility, also installed as part of the Co:Z toolkit, makes it easy to run **cozsftp** (or other Unix shell scripts) directly as z/OS batch jobs.

The authentication with the remote system must be set up so as not to require any user interaction. There are three ways to do this with OpenSSH:

- Use the SSH_ASKPASS environment variable to point to a program that will read a password.
- Use an OpenSSH public/private keypair.
- Use a RACF Digital Certificate.

For details on these three authentication options, see *Appendix F, Client Authentication Mechanisms*. Note that instructions in this appendix must be followed in order to run the examples described below.

Notes for running batch mode SFTP

When sftp is run in batch mode, it is important to know that sftp will abort if any of the supplied commands fail (i.e. complete with a non-zero return code). This behavior is different from an interactive sftp session, where a failed command will report an error, but the session will continue. In cases where a failed command is expected or acceptable (e.g. **rm old_file**, where old_file may not exist) it is useful to direct batch mode sftp to continue processing. To do this, prefix the command with a dash (-):

```
-rm old_file
```

Sample SFTPPROC and batch scripts

A sample **SFTPPROC** and batch scripts (installed in <COZ_HOME>/samples/sftp_batch) are distributed with the Co:Z toolkit to simplify maintenance and support of batch jobs using the Co:Z SFTP client. Using these samples achieves the following:

- **COZBATCH** customized for running CO:Z SFTP
- installation default options separated from individual JCL members
- standards defined for a set of variables controlling connection, authentication, options and filenames
- unix shell script logic separated into separate reusable script files

The reusable script files that are distributed with the Co:Z toolkit are the following:

- **sftp_get.sh** - Get a file from a remote system to a local (z/OS) file using the cozsftp command. This script connects a cozsftp client to a remote system running sshd and issues a get command to move a file from the remote system to a local file on z/OS. cozsftp transfer options (lzopts) can be specified to customize the transfer.
- **sftp_put.sh** - Put a local (z/OS) file to a remote system using the cozsftp command. This script connects a cozsftp

client to a remote system running sshd and issues a put command to move a local file to the remote system. cozsftp transfer options (lzopts) can be specified to customize the transfer.

- **sftp_connect.sh** - Connect to a remote system using the Co:Z toolkit cozsftp command. This script connects a cozsftp client to a remote system running sshd and prepares it to accept batch commands.
- **sftp_cat.sh** - Get multiple files from a remote system and concatenate them to a local (z/OS) file using the cozsftp command. This script connects a cozsftp client to a remote system running sshd and issues an ls command to get a list of files to get. Each of these files is then retrieved and written to the specified local file on z/OS. cozsftp transfer options (lzopts) can be specified to customize the transfer.

The standard set of variables to control connection, authentication, options and filenames are defined in the table below. Variables used by all scripts are required for sftp_connect.sh. sftp_cat.sh, sftp_get.sh, and sftp_put.sh invoke sftp_connect.sh to establish a connection with the remote host. Some variables are used only for specific scripts as noted in the Script column.

Table 4.1. Script Variables

Variable	Script	Required	Description
user	all	required	Set to the remote userid
host	all	required	Set to the remote host
port	all	optional	Set to the sshd port on the remote host. Port 22 is used by default
pwdsn	all	optional	Set to a fully qualified dataset name (or fully qualified dataset member) containing the user's remote system password. If so, SSH_ASKPASS authentication will be used. Note: If neither pwdsn or cert is set, z/OS OpenSSH defaults will be used for public/private key authentication.
cert	all	optional	Set to the name of a SAF digital certificate using one of the following formats: RING_NAME (no whitespace allowed; the default label will be used) RING_NAME:LABEL_NAME (no whitespace allowed) "RING_NAME LABEL_NAME" (whitespace between ring and label) If the first or second format is used, the connection will be authenticated using the Co:Z Toolkit saf-ssh-agent, which is the recommended approach as hardware private keys are supported. If the third format is used,

Variable	Script	Required	Description
			the z/OS OpenSSH identityKeyRingLabel option will be used for authentication. Note: If neither pwdsn or cert is set, z/OS OpenSSH defaults will be used for public/private key authentication.
sftp_opts	all	optional	Set to any desired SFTP options, including any ssh specific options (designated via the -o switch). Set using contatenation unless the prior variable setting is being overwritten.
cozbin_dir	all	optional	Set to the absolute path of the installed Co:Z Toolkit "bin" directory. If not set, this directory must be present in the current PATH environment variable.
script_dir	all	optional	Set to the absolute path of the directory containing the sftp_batch scripts (including this file). All scripts are assumed to be in this directory. If not set, this directory must be present in the current PATH environment variable.
lfile	sftp_get.sh and sftp_put.sh	required	Set to the local file to be created or transferred.
rfile	sftp_get.sh and sftp_put.sh	required	Set to the remote file to get or put.
lzopts	sftp_get.sh, sftp_put.sh, and sftp_cat.sh	optional	Set to cozsftp transfer options required for the transfer (e.g. mode=text,replace=no)
ldsn	sftp_cat.sh	required	Set to the local dataset or DD to be written
rpat	sftp_cat.sh	required	Set to the remote file pattern to get

Using the sample **SFTPPROC** and scripts, a batch job that gets a file from a remote host and stores it in a data set can be as simple as the following:

```
//PROCLIB JCLLIB ORDER='COZUSER.COZ.SAMPJCL'
//*
//*****************************************************************
//* Use the sftp_get.sh script to retrieve a remote file to a local
//* dataset.
//*****************************************************************
//SFTPGET EXEC PROC=SFTPPROC
//SFTPIN DD *
cert="MY-RING:RSA-CERT"
```

```
user=myuser
host=myhost
lzopts="mode=text"
lfile=//DD:MYDD
rfile=/etc/profile

. $script_dir/sftp_get.sh

//MYDD  DD DSN=COZUSER.SFTPGET.DATA,DISP=(MOD,KEEP),
//         DCB=(LRECL=80,RECFM=FB),SPACE=(CYL,(3,1))
//*
```

The sections below describe the **SFTPPROC**, installation default options, and a few more examples.

PROC for executing the Co:Z SFTP client (cozsftp) in batch

The **SFTPPROC** sample JCL distributed with the Co:Z toolkit can be used as a tailorable model for customizing COZBATCH for using CO:Z SFTP.

```
//*********************************************************************
//*
//* PROC for executing the Co:Z SFTP client (cozsftp) in batch
//*
//* Tailor the proc for your installation:
//* 1.) Tailor LIBRARY with the PDSE that contains the
//*     COZ load module.
//* 2.) Tailor SFTPIND= to point to SAMPJCL member that contains
//*     site specific shell variable settings for running the
//*     Co:Z SFTP batch scripts
//* 3.) Review the Co:Z SFTP batch scripts (located in
//*     $COZHOME/samples/sftp_batch) for additional shell variables
//*     to set for individual jobs to get, put, connect, etc...
//*********************************************************************
//EXSFTP    PROC ARGS=,                          < [-L<log_opt>] ❶
//   LIBRARY='COZUSER.COZ.LOADLIB',              < STEPLIB FOR COZBATCH
//   SFTPIND='COZUSER.COZ.SAMPJCL(SFTPIND)',     < Installation defaults ❷
//   REGSIZE='64M',                              < Execution region size
//   LEPARM=''
//RUNSFTP  EXEC PGM=COZBATCH,REGION=&REGSIZE, ❸
//   PARM='&LEPARM/&ARGS'
//STEPLIB  DD DSN=&LIBRARY,DISP=SHR
//STDIN    DD DSN=&SFTPIND,DISP=SHR ❹
//         DD DDNAME=SFTPIN
//SFTPIN   DD DUMMY                             < Customized stdin to SFTP ❺
//*
// PEND
```

❶ **COZBATCH** logging may be added to **ARGS** for problem diagnosis.

❷ Defines the member that contains the installation CO:Z SFTP defaults. These defaults can be overriden in

individual jobs as necessary.

❸ Defines the program to execute as **COZBATCH**, a utility similar to IBM's **BPXBATCH**. **COZBATCH** runs a Unix login shell in the original address space.

❹ Ensures that the site specific installation defaults are included first in STDIN, before any job specific commands.

❺ Defines a name for STDIN allowing jobs using this proc to include commands in STDIN.

Co:Z SFTP Batch Script Settings

The SFTPIND sample JCL member distributed with the Co:Z toolkit can be used as a tailorable model for CO:Z SFTP installation defaults.

```
################################################################################
# Co:Z SFTP Batch Script Settings
# The shell variables below can be set to site specific values, but may
# be overridden in individual jobs.
################################################################################

#
# CONFIGURATION VARIABLES:
#
# cozbin_dir - May be set to the absolute path of the installed Co:Z Toolkit
#              "bin" directory.  If not set, this directory must be present in
#              the current user's PATH environment variable.
# script_dir - May be set to the absolute path of the directory containing the
#              sftp_batch scripts (including this file).  All scripts are
#              assumed to be in this directory.  If not set, this directory must
#              be present in the current user's PATH environment variable.
#
cozbin_dir="/usr/local/coz/bin"   ❶
script_dir="/usr/local/coz/samples/sftp_batch" ❷

#
# SFTP OPTIONS VARIABLE:
#
# sftp_opts  - May be set to any site specific SFTP options, including any ssh
#              options (designated via the -o switch).
#
sftp_opts=""
sftp_opts="$sftp_opts -oConnectTimeout=60"
sftp_opts="$sftp_opts -oServerAliveInterval=60" ❸
#
# Set the following option to "no" if you would like to
# automatically accept host keys for new servers.
sftp_opts="$sftp_opts -oStrictHostKeyChecking=yes"
```

❶ Defines a variable for the location of the cozsftp executable. This variable is used by the sftp batch scripts to execute the cozsftp command.

❷ Defines a variable for the location of the sample or customized version of the sftp batch scripts. This variable is used in all jobs executing the sftp batch scripts.

❸ Sets global installation options as necessary. Note that the sftp_opts variable is appended as each option is added. Jobs using **SFTPPROC** can reset or append to these options using this variable.

Logging in batch

Co:Z SFTP Client logging is described earlier in this chapter (*Section 4.2, "Co:Z SFTP client logging"*). This section describes how to set these options in batch.

OpenSSH SFTP logging is enabled by passing the -v option on the sftp command. This can be done in batch by appending the desired -v option to the sftp_opts variable in the STDIN section of the job:

```
//SFTPIN DD *
sftp_opts="$sftp_opts -vvv"
```

Co:Z SFTP client logging is enabled by exporting the `COZ_LOG` variable in the STDIN section of the job.

```
//SFTPIN DD *
export COZ_LOG=T
```

COZBATCH logging is enabled by adding a parameter to the EXEC statement for the SFTPRPOC. The following example uses the `-LD` command switch to set the default logging level to "Debug" for COZBATCH. The `t` option is also used to prefix all messages with a timestamp.

```
//SFTPGET EXEC PROC=SFTPPROC,
        ARGS='-LD,t'
//SFTPIN DD *
```

Finally, it is often very helpful to see details about the Unix System Service shell script processing of the input to COZBATCH. This can be controlled by explicitly requesting a login shell (`/bin/sh -L`) along with the trace option (`-x`). The following example sets *both* the COZBATCH logging level to "Debug" and these shell options:

```
//SFTPGET EXEC PROC=SFTPPROC,
        ARGS='-LD,t /bin/sh -L -x'
//SFTPIN DD *
```

Batch job containing examples of running cozsftp in batch

The SFTPSAMP sample JCL distributed with the Co:Z toolkit can be used as a tailorable model for writing batch jobs using CO:Z SFTP.

```
//SFTPSAMP   JOB (),'DOVETAIL',MSGCLASS=H,NOTIFY=&SYSUID
//PROCLIB JCLLIB ORDER='COZUSER.COZ.SAMPJCL'
//*
//*****************************************************************
//*
//* Batch job containing examples of running cozsftp in batch
//*
//* Tailor the proc and job for your installation:
//* 1.) Modify the Job card per your installation's requirements
//* 2.) Modify the PROCLIB card to point to this PDS, or wherever
```

```
//*     the SFTPPROC procedure has been installed.
//*
//*********************************************************************
//*
//*********************************************************************
//* Use the sftp_get.sh script to retrieve a remote file to a local
//* dataset.  This example uses a user ssh key stored in a SAF
//* digital certificate
//*********************************************************************
//SFTPGET EXEC PROC=SFTPPROC ❶
//SFTPIN DD * ❷
cert="MY-RING:RSA-CERT" ❸
user=myuser
host=myhost
lzopts="mode=text"
lfile=//DD:MYDD
rfile=/etc/profile

. $script_dir/sftp_get.sh ❹

//MYDD  DD DSN=COZUSER.SFTPGET.DATA,DISP=(MOD,KEEP),
//         DCB=(LRECL=80,RECFM=FB),SPACE=(CYL,(3,1))
//*
//*********************************************************************
//* Use the sftp_put.sh script to send a local file to a remote
//* file.  This example uses a password (via the SSH_ASKPASS protocol)
//* to connect to the remote system
//*********************************************************************
//SFTPPUT EXEC PROC=SFTPPROC
//SFTPIN DD *
pwdsn="COZUSER.COZ.SAMPJCL(PW)"
user=myuser
host=myhost
lzopts="mode=text"
lfile=/etc/profile
rfile=/home/myuser/zprofile.txt

# Don't try to use our public key even if we have a default one
# This would not normally be a required setting
sftp_opts="$sftp_opts -oPubkeyAuthentication=no"

. $script_dir/sftp_put.sh

//*
//*********************************************************************
//* Use the sftp_cat.sh script to retrieve multiple files from a remote
//* system and concatenate them to a local dataset.  This example uses
//* z/OS OpenSSH defaults for public/private key authentication
//* (because neither the "cert" or "pwdsn" variables are defined)
//*********************************************************************
//SFTPCAT EXEC PROC=SFTPPROC
//SFTPIN DD *
user=myuser
```

```
host=myhost
lzopts="mode=text"
ldsn="//DD:MYDD"
rpat=/home/myuser/doc/*.txt

. $script_dir/sftp_cat.sh

/*
//MYDD  DD DSN=COZUSER.SFTPCAT.DATA,DISP=(MOD,KEEP),
//         DCB=(LRECL=80,RECFM=FB),SPACE=(CYL,(3,1))
//*
//*****************************************************************
//* Use the sftp_connect.sh script to connect to a remote system
//* and send customized sftp commands.
//*****************************************************************
//SFTPCONN EXEC PROC=SFTPPROC
//SFTPIN DD *
cert="MY-RING:RSA-CERT"
user=myuser
host=myhost

. $script_dir/sftp_connect.sh << EOB ❺
ls -al
EOB

//
```

❶ Each step in this sample job uses the **SFTPPROC**.
❷ Each step uses SFTPIN defined by the **SFTPPROC**.
❸ Sets all variables required for sftp_get.sh. Note that some required variables are in the installation global defaults. Global variables may be optionally overriden here before calling the shell script.
❹ Executes sftp_get.sh to get a file from the remote system saving it in the data set defined by //MYDD.
❺ The symbols '<< EOB' followed by an ending 'EOB' define a Here document which, in this example, is an inline string containing a SFTP command.

Wild-card downloading using a DD

In *release 2.4.0*, support was added to allow multiple files to be downloaded using a wild-card pattern (*) to a single DD if the DD was allocated with DISP=MOD.

```
//SFTPGET EXEC PROC=SFTPPROC
//SFTPIN DD *
user=myuser
host=myhost
pwdsn="COZUSER.COZ.SAMPJCL(PW)"
lzopts="mode=text"
lfile=//DD:MYDD       ❶
rfile=/somedir/*.trn  ❷
. $script_dir/sftp_get.sh  ❸

//MYDD DD DSN=COZUSER.SFTP.MULTIGET.DATA,DISP=(MOD,CATLG,DELETE), ❶
//        DCB=(LRECL=2052,RECFM=FB),SPACE=(CYL,(3,1))
```

❶ The `lfile` variable references a DD in the job step that is allocated with DISP=MOD. This allows multiple files to be downloaded, in succession, to the end of the same target dataset. Each matching file will be downloaded separately, in alphabetical order.

❷ The **get** command uses a wild-card (*) pattern to select any file in the `/somedir` directory that ends in `".trn"`.

❸ The underlying **get** subcommand generated by the **sftp_get.sh** script will be:
get /somedir/*.trn //DD:MYDD

5. Automation with System Console Messages

If console messages are needed for automation, the following methods can be used:

5.1 Console Notification Co:Z SFTP Option (Notify)

By setting the Co:Z SFTP `notify` option, WTO messages can be written on completion of the following commands: **put**, **get**, **rm** and **rename**. These command completion messages are supported by the Co:Z SFTP client and server. The message text written to the sytem console is configurable in the sitewide server and client configuration files. See *Section C.1, "Specifying notification (immutable) options"* for information on the message template configuration and *Section B.1, "General transfer options"* for information on the `notify` option.

5.2 Post Completion Exit (CZPOSTPR)

The Co:Z SFTP server supports a user post completion exit (CZPOSTPR) which is compatible with the FTPOSTPR exit supported by the IBM z/OS FTP server. A customer may implement this exit to write WTO messages as required for an installation's automation needs. For additional information, see the *Co:Z SFTP Exits Guide*.

5.3 SMF Exit

Co:Z SFTP writes SMF type 119 records that capture file transfer completion details. A customer may implement a SMF exit to write WTO messages as required for an installation's automation requirements. For additional information on the records written by the Co:Z SFTP server and client, see *Appendix E, SMF Information*.

Appendix A. Command Reference

- *cozsftp(1)*
- *sftp-server(1)*

Name

cozsftp — secure file transfer program for z/OS

Synopsis

```
cozsftp [-1246Cpqrv] [-B buffer_size]
        [-b batchfile] [-c cipher]
        [-D sftp_server_path] [-F ssh_config]
        [-i identity_file] [-k my-ring:my-cert] [-l limit]
        [-o ssh_option] [-P port]
        [-R num_requests] [-S program]
        [-s subsystem | sftp_server] host
cozsftp [user@]host[:file ...]
cozsftp [user@]host[:dir[/]]
cozsftp -b batchfile [user@]host
```

Description

cozsftp is an interactive file transfer program, similar to ftp(1), which performs all operations over an encrypted ssh(1) transport. It may also use many features of ssh, such as public key authentication and compression. sftp connects and logs into the specified host, then enters an interactive command mode.

The second usage format will retrieve files automatically if a non- interactive authentication method is used; otherwise it will do so after successful interactive authentication.

The third usage format allows sftp to start in a remote directory.

The final usage format allows for automated sessions using the -b option. In such cases, it is necessary to configure non-interactive authentication to obviate the need to enter a password at connection time (see sshd(8) and ssh-keygen(1) for details).

Since some usage formats use colon characters to delimit host names from path names, IPv6 addresses must be enclosed in square brackets to avoid ambiguity.

z/OS specific notes:

- cozsftp is a z/OS specific version of the OpenSSH **sftp** command
- Includes support for z/OS artifacts such as data sets and JES spool access
- Features and options that are different from the base **sftp** command are noted below with [**z/OS ...**]

Options

`-1`
Specify the use of protocol version 1.

`-2`

Specify the use of protocol version 2.

`-4`
Forces cozsftp to use IPv4 addresses only.

`-6`
Forces cozsftp to use IPv6 addresses only.

`-B` buffer_size
Specify the size of the buffer that cozsftp uses when transferring files. Larger buffers require fewer round trips at the cost of higher memory consumption. The default is 32768 bytes.

`-b` batchfile
Batch mode reads a series of commands from an input batchfile instead of stdin. Since it lacks user interaction it should be used in conjunction with non-interactive authentication. A batchfile of `-' may be used to indicate standard input. cozsftp will abort if any of the following commands fail: get, put, reget, rename, ln, rm, mkdir, chdir, ls, lchdir, chmod, chown, chgrp, lpwd, df, symlink, and lmkdir. Termination on error can be suppressed on a command by command basis by prefixing the command with a `-' character (for example, -rm /tmp/blah*).

`-C`
Enables compression (via ssh's -C flag).

`-c` cipher
Selects the cipher to use for encrypting the data transfers. This option is directly passed to ssh(1).

`-D` sftp_server_path
Connect directly to a local sftp server (rather than via ssh(1)). This option may be useful in debugging the client and server.

`-F` ssh_config
Specifies an alternative per-user configuration file for ssh(1). This option is directly passed to ssh(1).

`-i` identity_file
Selects the file from which the identity (private key) for public key authentication is read. This option is directly passed to ssh(1).

`-k` my-ring:my-cert
[**z/OS only**] Specifies a SAF/RACF certificate for SSH authentication. if **:mycert** is not specified, the default certificate label for the given key ring will be used.

`-l` limit
Limits the used bandwidth, specified in Kbit/s.

`-o` ssh_option
Can be used to pass options to ssh in the format used in ssh_config(5). This is useful for specifying options for which there is no separate sftp command-line flag. For example, to limit the number of password prompts allowed to one use: cozsftp -oNumberOfPasswordPrompts=1. For full details of the options allowed, and their possible values, see ssh_config(5).

`-P` port
Specifies the port to connect to on the remote host.

`-p`

[**z/OS Unix files only**] Preserves modification times, access times, and modes from the original files transferred.

`-q`
Quiet mode: disables the progress meter as well as warning and diagnostic messages from ssh(1).

`-R` num_requests
Specify how many requests may be outstanding at any one time. Increasing this may slightly improve file transfer speed but will increase memory usage. The default is 64 outstanding requests.

`-r`
[**z/OS Unix files only**] Recursively copy entire directories when uploading and downloading. Note that sftp does not follow symbolic links encountered in the tree traversal.

`-S` program
Name of the program to use for the encrypted connection. The program must understand ssh(1) options.

`-s` subsystem | sftp_server
Specifies the SSH2 subsystem or the path for an sftp server on the remote host. A path is useful for using sftp over protocol version 1, or when the remote sshd(8) does not have an sftp subsystem configured.

`-v`
Raise logging level. This option is also passed to ssh.

Interactive Commands

Once in interactive mode, cozsftp understands a set of commands similar to those of ftp(1). Commands are case insensitive. Pathnames that contain spaces must be enclosed in quotes. Any special characters contained within pathnames that are recognized by glob(3) must be escaped with backslashes (`\').

ascii
[**z/OS added**] Alias for: lzopts mode=text

append [-Pp] local-path [remote-path]
[**z/OS added**] Upload local-path and append it to the end of a remote file on the remote machine. If the remote path name is not specified, it is given the same name it has on the local machine. local-path may contain glob(3) characters and may match multiple files. If it does and remote-path is specified, then remote-path must specify a directory. The remote file is created if it does not exist.

[**z/OS Unix files only**] If either the -P or -p flag is specified, then full file permissions and access times are copied to the remote file.

[**z/OS Note:**] The append command is implemented by opening the remote file and then obtaining its current size as the initial offset for writing data. Some SFTP server products may not support this correctly. If the remote server is Co:Z SFTP, then the append command will be rejected if the current file transfer options (e.g. linerule, translation) allow the file size to be changed. The append command is not supported to a remote Co:Z SFTP data set; use the **put** command after setting the server option: **ls /+disp=mod** to append to the end of a remote z/OS data set.

binary
[**z/OS added**] Alias for: lzopts mode=binary

bye
Quit cozsftp.

cd path
Change remote directory to path.

chgrp grp path
[**z/OS Unix files only**] Change group of file path to grp. path may contain glob(3) characters and may match multiple files. grp must be a numeric GID.

chown own path
[**z/OS Unix files only**] Change owner of file path to own. path may contain glob(3) characters and may match multiple files. own must be a numeric UID.

df [-hi] [path]
[**z/OS Unix files only**] Display usage information for the filesystem holding the current directory (or path if specified). If the -h flag is specified, the capacity information will be displayed using "human-readable" suffixes. The -i flag requests display of inode information in addition to capacity information. This command is only supported on servers that implement the ``statvfs@openssh.com'' extension.

exit
Quit cozsftp.

get [-aPpr] remote-path [local-path]
Retrieve the remote-path and store it on the local machine. If the local path name is not specified, it is given the same name it has on the remote machine. remote-path may contain glob(3) characters and may match multiple files. If it does and local-path is specified, then local-path must specify a directory.

If the -a flag is specified, then attempt to resume partial transfers of existing files. Note that resumption assumes that any partial copy of the local file matches the remote copy. If the remote file differs from the partial local copy then the resultant file is likely to be corrupt.

If either the -P or -p flag is specified, then full file permissions and access times are copied too.

If the -r flag is specified then directories will be copied recursively. Note that sftp does not follow symbolic links when performing recursive transfers.

[**z/OS Unix files only**] No flags are supported if either the local or remote file is a z/OS data set. The -a is also not supported for z/OS Unix files if the current file transfer options (e.g. linerule, translation) allow the file size to be changed

help
Display help text.

lcd path
Change local directory to path.

lls [ls-options [path]]
Display local directory listing of either path or current directory if path is not specified. ls-options may contain any flags supported by the local system's ls(1) command. path may contain glob(3) characters and may match multiple files.

[**z/OS**] lls -h and -S are not supported because the z/OS ls command does support these options. The -n option is displayed as -l for data sets rather than displaying a numerical listing.

lmkdir path
[**z/OS Unix files only**] Create local directory specified by path.

ln [-s] oldpath newpath
[**z/OS Unix files only**] Create a link from oldpath to newpath. If the -s flag is specified the created link is a symbolic link, otherwise it is a hard link.

lpwd
Print local working directory.

ls [-1afhlnrSt] [path]
Display a remote directory listing of either path or the current directory if path is not specified. path may contain glob(3) characters and may match multiple files.

The following flags are recognized and alter the behaviour of ls accordingly:

`-1`
Produce single columnar output.

`-a`
List files beginning with a dot (`.').

`-f`
Do not sort the listing. The default sort order is lexicographical.

`-h`
When used with a long format option, use unit suffixes: Byte, Kilobyte, Megabyte, Gigabyte, Terabyte, Petabyte, and Exabyte in order to reduce the number of digits to four or fewer using powers of 2 for sizes (K=1024, M=1048576, etc.).

If the remote server is Co:Z SFTP, the numerical long form listing is displayed data sets when -h is used with either -l or -n

`-l`
Display additional details including permissions and ownership information.

If the remote server is Co:Z SFTP, for data sets this returns the following (unless the Co:Z **unixls** option is set): Volume, Referred, Ext, Tracks, Used, Recfm, Lrecl, BlkSz, Dsorg, Dsname.

`-n`
Produce a long listing with user and group information presented numerically.

If the remote server is Co:Z SFTP, for data sets this displays the estimated data set size when possible.

`-r`
Reverse the sort order of the listing.

`-S`
Sort the listing by file size.

`-t`
Sort the listing by last modification time.

lumask umask
Set local umask to umask.

lzopts [-a] [option=value,...]
[**z/OS added**] Set local (client) file transfer options. These options are set prior to initiating file/dataset transfers from z/OS to a remote host.

Multiple options can be set by separating the option=value pairs with commas. An error is returned if one or more of the options was incorrectly specified, but the remaining options are set as requested.

The active options and their settings can be displayed by issuing the **lzopts** command without arguments. The -a option can be specified to list all available options, even those that are not active.

For the set of Co:Z SFTP Options, see the *Co:Z SFTP - User's Guide*.

mkdir path
Create remote directory specified by path. If the remote server is Co:Z SFTP, the path may be a //data-set-name in which case Co:Z SFTP serve will create a PDS or PDSE if no data set with this name alread exists.

progress
Toggle display of progress meter.

put [-Ppr] local-path [remote-path]
Upload local-path and store it on the remote machine. If the remote path name is not specified, it is given the same name it has on the local machine. local-path may contain glob(3) characters and may match multiple files. If it does and remote-path is specified, then remote-path must specify a directory.

If either the -P or -p flag is specified, then full file permissions and access times are copied too.

If the -r flag is specified then directories will be copied recursively. Note that sftp does not follow symbolic links when performing recursive transfers.

[**z/OS Unix files only**] No flags are supported if either the local or remote file is a z/OS data set. The -a is also not supported for z/OS Unix files if the current file transfer options (e.g. linerule, translation) allow the file size to be changed

pwd
Display remote working directory.

quit
Quit cozsftp.

reget [-Ppr] remote-path [local-path]
[**z/OS Unix files only**] Resume download of remote-path. Equivalent to get with the -a flag set. **Note:** reget is not supported if the current file transfer options (e.g. linerule, translation) can cause the file size to be changed.

rename oldpath newpath
Rename remote file from oldpath to newpath.

rm path
: Delete remote file specified by path.

rmdir path
: [**z/OS Unix files only**] Remove remote directory specified by path.

symlink oldpath newpath
: [**z/OS Unix files only**] Create a symbolic link from oldpath to newpath.

version
: Display the sftp protocol version.

zopts [-a] [option=value,...]
: [**z/OS added**] Set server file transfer options, if the server is a Co:Z SFTP server. These options are set prior to initiating file/dataset transfers from z/OS to a remote host.

  The zopts command is functionally equivalent to the ls /+[option=value,...] command used by existing clients to set Co:Z sftp-server file transfer options.

  Multiple options can be set by separating the option=value pairs with commas. An error is returned if one or more of the options was incorrectly specified, but the remaining options are set as requested.

  The active options and their settings can be displayed by issuing the **zopts** command without arguments. The -a option can be specified to list all available options, even those that are not active.

  For the set of Co:Z SFTP Options, see the *Co:Z SFTP - User's Guide*.

!command
: Execute command in local shell.

!
: Escape to local shell.

?
: Synonym for help.

See Also

ftp(1), ls(1), scp(1), sftp(1), ssh(1), ssh-add(1), ssh-keygen(1), glob(3), ssh_config(5), sftp-server(8), sshd(8).

T. Ylonen and S. Lehtinen, SSH File Transfer Protocol, draft-ietf-secsh- filexfer-00.txt, January 2001, work in progress material.

Name

sftp-server — SFTP server subsystem

Synopsis

```
sftp-server [-ehR] [-d start_directory] [-f log_facility] [-l log_level] [-u umask]
```

Description

Note

When using Co:Z SFTP, this command is not invoked directly, it is invoked using the shell script **sftp-server.sh**.

sftp-server is a program that speaks the server side of SFTP protocol to stdout and expects client requests from stdin. sftp-server is not intended to be called directly, but from sshd(8) using the Subsystem option.

Command-line flags to sftp-server should be specified in the Subsystem declaration. See sshd_config(5) for more information.

z/OS specific notes:

- Features and options that are different from the base **sftp-server** command are noted below with [**z/OS**]

Options

`-d` start_directory
: specifies an alternate starting directory for users. The pathname may contain the following tokens that are expanded at runtime: %% is replaced by a literal '%', %d is replaced by the home directory of the user being authenticated, and %u is replaced by the username of that user. The default is to use the user's home directory. This option is useful in conjunction with the sshd_config(5) ChrootDirectory option.

  [**z/OS feature**] The SFTP_ZOS_INITIAL_DIR environment variable, if set, will override the start_directory set by this option. See Configuring the Co:Z SFTP Server in the *Co:Z SFTP - User's Guide*.

`-e`
: Causes sftp-server to print logging information to stderr instead of syslog for debugging.

`-f` log_facility
: Specifies the facility code that is used when logging messages from sftp-server. The possible values are: DAEMON, USER, AUTH, LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, LOCAL7. The default is AUTH.

`-h`
: Displays sftp-server usage information.

`-l` log_level
: Specifies which messages will be logged by sftp-server. The possible values are: QUIET, FATAL, ERROR, INFO, VERBOSE, DEBUG, DEBUG1, DEBUG2, and DEBUG3. INFO and VERBOSE log transactions that sftp-server performs on behalf of the client. DEBUG and DEBUG1 are equivalent. DEBUG2 and DEBUG3 each specify higher levels of debugging output. The default is ERROR.

`-R`
: Places this instance of sftp-server into a read-only mode. Attempts to open files for writing, as well as other operations that change the state of the filesystem, will be denied.

`-u` umask
: Sets an explicit umask(2) to be applied to newly-created files and directories, instead of the user's default mask.

On some systems, sftp-server must be able to access /dev/log for logging to work, and use of sftp-server in a chroot configuration therefore requires that syslogd(8) establish a logging socket inside the chroot directory.

See Also

sftp(1), ssh(1), sshd_config(5), sshd(8).

T. Ylonen and S. Lehtinen, SSH File Transfer Protocol, draft-ietf-secsh-filexfer-02.txt, October 2001, work in progress material.

Appendix B. Co:Z SFTP options

B.1 General transfer options

The following table describes the general transfer options. The usage columns describe when the option (if active) will apply during transfer. Usage *Read* means using Co:Z SFTP (server or client) to read a z/OS file. Usage *Write* means using Co:Z SFTP (server or client) to write a z/OS file.

Note

Options that have a blank *value* column are on/off options. They are activated by supplying the option name by itself (no values allowed) and deactivated by prefixing the option name with the prefix `NO`. For example: `trim` and `notrim`.

Table B.1. General transfer options

Keyword		Usage			
Name	**Value**	**Datasets**	**POSIX**	**Read**	**Write**
clientcp	<codepage>	X	X	X	X
estsize		X	X	X	
jesincsysin		X		X	
linerule	`cr | crlf | crnl | l4 | lf | nl | rdw | mfrdw | flexible | 0xbb[bb..] | none`	X	X	X	X
mode	`binary | text`	X	X	X	X
notify		X	X	X	X
overflow	`error | flow | trunc | wrap`	X			X
pad	`<pad_char> | 0xbb`	X		X	X
replace		X	X		X
servercp	<codepage>	X	X	X	X
technique	<technique_string>	X	X	X	X
trtab	`STANDARD | translate_table_dsname`	X	X	X	X
trim		X		X	

`clientcp`
: Specifies the name of the client codepage used when performing text mode transfers. Data will be converted between the server codepage (servercp) and this code page. The codepage must either be a single byte codepage or any multi-byte codepage that has single-byte line terminators (e.g. UTF-8). The z/OS Unix command **iconv -l** lists the available codepages. For the Co:Z SFTP server the default is `ISO8859-1`. For the Co:Z SFTP client, the default is the default locale codeset of the client's process (some variant of EBCDIC).

  A TranslationException is logged when the transferred file is malformed. When possible, the offset of the last successfully translated byte is captured in the error log message.

`estsize`
: When **estsize** (the default) is enabled, Co:Z SFTP returns the actual size for Unix files and an estimated size for MVS data sets. When **estsize** is disabled (**NOestsize**), the size returned is zero and `SSH_FILEXFER_ATTR_SIZE` is off indicating that the attribute size is not present. This has the effect of telling the client on a file read that there is no estimated size.

`jesincsysin`
: When option `jesincsysin` is enabled, concatenated spool file downloads will include SYSIN spool files. In addition, the separator between spool files will be annotated with the [step.[procstep.]]ddname. This option is a convenient way to download all job input and output.

`linerule`
: The values `cr`, `crlf`, `crnl`, `lf`, `nl` specify that, for text mode transfers, lines will be terminated with the given characters in the client codepage.

  `flexible` may be used when writing to files or datasets to indicate that any combination of cr, lf, or newline will be recognized as a line terminator.

  `rdw` specifies that IBM-style RDWs are used as prefixes. `l4` specifies that each record is delimited (preceded) by a four byte length of the record that follows. Note: Unlike the `rdw` option, this length value does **not** include the size of the length field. `mfrdw` indicates that Micro-focus file and records headers are used.

  `0xbb[bb..]` may be used to specify a sequence of one or more bytes in the source codepage. `none` should be used when no line terminators are to be recognized/used.

  The default is `flexible` for writing and **lf** for reading.

`mode`
: Specifies whether transfers are as-is (binary) or subject to codepage/linerule/overflow/pad processing (text). The default is `binary`.

`notify`
: Specifies whether a message should be written to the console on completion of a put or get command by the Co:Z SFTP server and client. Notifications are also written on completion of a remove or rename command on the server. The default is `NOnotify`. In order to enable this option, a valid message template must be defined in the site-wide server or client configuration. For addition information, see *Section C.1, "Specifying notification (immutable) options"*. Notifications are disabled if the Co:Z SFTP server log file has been redirected to `/dev/console` due to the potential for recursive logging.

`overflow`
: For text-mode dataset write processing, controls the treatment of lines longer than the maximum dataset record length. The default is `wrap`. When set to `error`, an error is returned if the source line is longer than the maximum record length. When set to `flow`, source lines longer than the maximum record length are flowed across subsequent records. For fixed record formats, the pad character is used to complete the final record resulting from the source line. When set to `trunc`, source lines longer than the maximum record length are truncated. When set to `wrap`, source lines longer than the maximum record length are broken into multiple records.

`pad`
: For text-mode dataset write processing, specifies the character to use when padding lines into fixed-length dataset records. For text-mode dataset read processing, this character also identifies the character to be trimmed if the `trim` is enabled. If given as `0xbb`, it specifies (in hex) a single-byte character in the source codepage. If not specified, the default is a space character in the local z/OS codepage.

`replace`
: This setting allows for existing datasets or files to be replaced. The default, if not specified, is `replace`, which allows for replacement. `NOreplace` can be set to prevent an existing dataset or file from being replaced.

  When using cozsftp, setting `NOreplace` with `lzopts` only applies to get commands.

  `NOreplace` for a put command requires that the server be running Co:Z SFTP Server. For other SFTP servers, this option is not supported.

  Note: If `NOreplace` is set, you may not create PDS members, regardless of whether the member exists, and you may also not write to GDG datasets using a positive (+n) relative reference.

`servercp`
: Specifies the name of the server codepage used when performing text mode transfers. Data will be converted between the client codepage (clientcp) and this code page. For the Co:Z SFTP server the default is default locale codeset of the server's process (some variant of EBCDIC). For the Co:Z SFTP client the default is `ISO8859-1`.

  A TranslationException is logged when the transferred file is malformed. When possible, the offset of the last successfully translated byte is captured in the error log message.

`technique`
: Specifies the Codepage conversion technique string. Used to override the default Unicode Services value of `LMREC`. For more information, see IBM's Unicode Services User's Guide and Reference (SA22-7649).

`trtab`
: Specifies the translate table to use for text mode transfers. This option overrides the `clientcp/servercp/technique` options if also given. If `STANDARD`, the translate table TCPIP.STANDARD.TCPXLBIN is used. If a dataset name is supplied, it is expected to be in the format produced by the TSO CONVXLAT command. Only single byte translations are supported. Specifically, the dataset DCB must be LRECL=256,RECFM=F and contain two translation table records. The first record is an ASCII-to-EBCDIC mapping; the second record is an EBCDIC-to-ASCII mapping. Additional comment records (starting with * in the first column) are allowed.

`trim`
: For text-mode dataset read transfers, enabling this options will cause pad characters to be trimmed from the

dataset records as they are read. The default, if not specified, is `trim`.

B.2 Miscellaneous options

The following table describes the miscellaneous options. These options do not apply to transfer operations, but affect the behavior of Co:Z SFTP.

Table B.2. Miscellaneous options

Name	Value	Notes
interimlogging	interval[.log | .sock | .both]	Sets the interval in seconds for logging interim messages and/or records for the progress of a file transfer. By default, the interval is set to zero (disabling interim logging). If this option is specified with a positive integer value for `interval` an interim record will be written *both* to the real-time Co:Z SMF API and as a summary message to the Co:Z client or server log approximately every `interval` seconds. The interval option may optionally be followed by one of the following: `.log` - interval log messages are only written to the Co:Z SFTP log file (stderr). For the Co:Z SFTP server, this is normally routed to the session log file. The log message is an (I)nformational message, if written, so the loglevel must be set to I or a more detailed level for it to be seen). `.sock` - interval log records are only written to the Co:Z SMF real-time interface socket. If this Unix domain socket (a Unix path) has not been created, then it is ignored. Refer to *Using the Real-Time Co:Z SMF Interface* for additional information on the real-time interface. `.both` - interval logs messages and records will both be recorded. This is the default sub-option if not specified. Note: When Co:Z SFTP client is running interactively (not in batch), the Progress meter must be disabled (using the `progress` subcommand) in order to write interim records or log messages. The recommended interval is two minutes or more. For example, to set a 3 minute interval, use: `interimlogging=180`
loglevel	`E|W|N|I|D|T|F`	Sets the logging level of the Co:Z sftp-server. The UPPER CASE values correspond to the list: (Error, Warning, Notice, Info, Debug, Trace, Fine).
maxcscnt	`nnnn`	Sets the maximum number of CatalogSearch entries returned for a given dataset level search. The default is 2000. *Use*

Name	Value	Notes
		caution when specifying a large maximum value, as it may cause Co:Z SFTP to run out of memory.
reqexits	`exit[.exit]...`	(server only) For each required exit listed, the corresponding loadmodule must be available and loaded. If not, an error message will be written to the log and the server session immediately terminated. The exit names that may be specified are: `CZCHKCMD`, `CZCHKIP`, `CZCHKPWD`, and `CZPOSTPR`. Installations that use exits will typically set this option in the `fixed:` section of `/etc/ssh/cozsftp_server_config`. Additionally, users should not have write access to individual sftp-server.rc files in order to prevent users from overriding installation exits with their own exits. See the *Sitewide server configuration* section for information on managing individual sftp-server.rc files.
servertimeout	`nnnn`	(server only) Sets a server side timeout value where nnnn is minutes. If the client is inactive longer than nnnn minutes, the server terminates the session. By default, no timeout value is configured. To configure this option globally, set this option in the fixed: section of /etc/ssh/cozsftp_server_config. This option may not be set or changed once the session has started.
showall		If active, all options will be shown on option display (**ls /+**). Inactive options are shown with a prefix of `NO`.
smf	U83 | U84	If active (the default, smf=U84), SMF 119 records will be written and IEFU84 called (when available) for file transfer events. If a failure occurs because SMF is disabled, this option will be automatically set to NOsmf and no further attempts will be made. To completely disable SMF recording supply "nosmf" in one of the config files: *Appendix C, Session config files*. This option may not be set or changed after the session has started.
ssh-le-options		(Co:Z SFTP client only) This option may be used to supply z/OS Language Environment options for the z/OS OpenSSH **ssh** command when it is invoked by **cozsftp**. This option may not be set or changed once the session has started. Referring to IBM APAR OA34819, we suggest that customers adopt the value for this setting that is in the sample `cozsftp_config` file supplied with the distribution.
unixls		If active (NOT the default), a UNIX long form listing is

Name	Value	Notes
		returned on client requests for directory listings of data sets. This format is incorrectly expected by many clients, particularly GUI clients, that do not fully adhere to the SFTP specification (refer to *Client Compatibility*). For many clients, setting this option • improves directory listings of data sets by identifying partition data sets as directories enabling double click to navigate to the members • shows jobs displayed by //-JES as directories enabling double click to navigate to the spool files • lists /+ output correctly For clients where /+ does not work by default, the option must be enabled in one of the config files, *Appendix C, Session config files*. Values returned in the listing for Owner, Group and Access are generally placeholder values because they are not available on z/OS. A dash will be displayed for Owner and Group. Access will be displayed as "-rw-r--r--" for files and "drw-r--r--" for directories. Last Modified is returned as "Jan 1 1970" when a valid date is not available. Some clients display this date; however, some display a date with the year 1899.

B.3 Dataset allocation options

The following table describes options that apply when transferring MVS datasets. The z/OS BPXWDYN service is used for dataset allocation and these options correspond to keywords available with BPXWDYN with similar syntax except that:

- `keyword=value` is used rather than `keyword(value)`
- periods are used in place of commas
- other minor differences as described below

The usage columns below describe when the option (if active) will apply during dataset transfer (none of these options, with the exception of `conddisp`, apply during POSIX file transfers). For more information on BPXWDYN, see *Using REXX and z/OS UNIX System Services - SA22-7806*

Note

The `conddisp` option is not a BPXWDN keyword, but is supported by Co:Z SFTP to handle dataset / POSIX file deletion in the result of a transfer error. If this option is set to `delete`, Co:Z SFTP server will attempt to delete any file or dataset that is being written to (on the client or the server) if the transfer is interrupted. In the OpenSSH sftp client, a Ctrl-C (SIG-INT) is caught in the client and it just closes the file, so there is no way for the server to see this as an interruption. In this case, it is still the client's responsibility to clean up the file.

Table B.3. BPXWDYN options

Keyword		Usage			
Name	**Value**	**Read**	**Write New**	**Write Existing**	**Sysout**
blksize	<numeric>		X		X
bufno	<numeric>	X	X	X	
conddisp	`catlg | delete`	X	X	X	X
copies	<numeric>				X
dataclas	<alphanum>		X		
dest	`dest[.user]`				X
dir	<numeric>		X		
disp	`old | shr | mod | new`	X	X	X	
dsntype	`library | pds | large | extreq | extpref | basic`		X		
dsorg	`ps | po | da`		X		
forms	<alphanum>				X

Keyword		Usage			
Name	**Value**	**Read**	**Write New**	**Write Existing**	**Sysout**
gdgnt			X		
hold					X
label	`nl | sl | nsl | sul | blp | ltm | al | aul`		X		
like	<Dataset Name>		X		
lrecl	<numeric>		X		X
maxvol	<numeric>		X		
mount		X	X	X	
mgmtclas	<alphanum>		X		
norecall		X		X	
outdes	<alphanum>				X
recfm	<alphanum>		X		X
release		X	X	X	
retpd	<numeric>		X		
sequence	<numeric>		X		
space	`blk.primary[.secondary] | trk.primary[.secondary] | cyl.primary[.secondary]`		X		
spin	unalloc				X
storclas	<alphanum>		X		
sysout	<sysout_class>				X
trtch	`noncomp | comp | c | e | et | t`		X		
ucount	<numeric>	X	X	X	
unit	<alphanum>		X		
vol	<alphanum>		X		
writer	<alphanum>				X

Appendix C. Session config files

The files `/etc/ssh/cozsftp_config` and `/etc/ssh/cozsftp_server_config` can be used to customize the options available for Co:Z SFTP client and server sessions respectively. The permissions for each of these files should be `0644`.

User and site-wide samples of both of these files are located in the `<COZ_INST>/samples` directory. These samples may be customized and placed at the above locations to make them active.

Each file has the sections `notification:`, `fixed:`, `default:` and `pattern:` which are described below. Additionally, individual users can provide their own file patterns and defaults (but not fixed options) in copies of these files in `$HOME/.ssh`

The individual Co:Z SFTP client and server options are described in: *Appendix B, Co:Z SFTP options*.

C.1 Specifying notification (immutable) options

Use the `notification:` section to specify site-wide message properties that *cannot* be overridden by individual users. When the `notify` Co:Z SFTP option is enabled, a message is written to the console on completion of put and get commands by the Co:Z SFTP server and client. Notifications are also written on completion of remove and rename commands on the server. For additional information on the `notify` option see *Appendix B, Co:Z SFTP options*. The table defines the configurable message properties.

Table C.1. Notification message properties

Property	Description
messageid	Specifies the message id assigned on successful command completion. By default, the message identifier is COZSC0001I on the client and COZSS0001I on the server.
errormessageid	Specifies the message id assigned on failed command completion. By default, the message identifier is COZSC0002E on the client and COZSS0002E on the server.
routingcodes	A comma separated list of routing codes. By default, none are specified. Invalid routing codes are ignored with a warning message written to the session log.
descriptorcodes	A comma separated list of descriptor codes. By default, none are specified. Invalid descriptor codes are ignored with a warning message written to the session log.
template.n	A line in the message template where n is a number from 1 to 9. This property can be used to specify up to 9 message lines. The message template contains variables that are replaced at runtime with information about the completed command. A variable is represented in the template with the following syntax: ${variable}. See the table below for the supported variables. Note that the

Property	Description
	characters $, {, and } are reserved and may not be used in message text. If a single line message is configured, the message may be 126 characters. If a multi-line message is configured, each line in the message is limited to 70 characters. If a line exceeds these limits, the line is truncated and the last character in the line is set to '*'.

The table below defines case sensitive variables that can be used in the message template.

Table C.2. Message template variables

Variable	Description
user	The user ID
remote_ip	The remote IP address
remote_port	The remote port
local_ip	The local IP address
local_port	The local port
cmd	The FTP command code. One of STOR, RETR, RNTO, or DELE.
comp_code	The file transfer completion code. One of 0, 4, or 12. Completion code 12 is set when a transfer is terminated before completion.
conddisp	The current CONDDISP setting: C for catalog, D for delete.
file	The fully qualified file name. A file with a leading slash is a POSIX file. A file without a leading slash is a data set.
file_type	One of SEQ or JES.
conf	The confidence level. Blank when not a transfer, i.e. remove or rename. 'unknown' for out bound transfers (RETR on the server, STOR for the client). 'high' for successful in bound transfers. 'low' when an error occurs on an in bound transfer.
kbs	Total bytes transfered rounded to the nearest 1024 bytes.
bs	Total bytes transfered
reply_code	FTP reply code
reply_string	FTP reply string. Note that this field is often longer than 70 characters in command failure scenarios; therefore, will be truncated in a multi-line message. When configured, it is recommended that this variable is the only text on the template

Variable	Description
	line.
session_id	A unique id assigned to the user's session. The session id is generated from the jobname followed by the last 5 digits of the pid.

Using the example configuration below,

```
notification:
#messageid=COZSS0001I
#errormessageid=COZSS0002E
#routingcodes=
#descriptorcodes=
template.1=${user},${remote_ip},${cmd},${comp_code},${reply_code}
template.2=${file}
template.3=${bs} bytes transferred
template.4=${reply_string}
```

the following message is written to the console on successful completion of a put command to the Co:Z SFTP server:

```
COZSS0001I COZUSER,192.28.145.64,STOR,0,250
/u/vendor/cozuser/testfile.txt
7956480 bytes transferred
Transfer completed successfully.
```

C.2 Specifying fixed (immutable) options

Use the `fixed:` section to specify site-wide options that *cannot* be overridden by individual users. Multiple options may be specified on a single line if separated by commas.

In the example below, the `smf` option is activated for all users, and because it it fixed, may not overridden by any user.

```
fixed:
smf
```

C.3 Specifying default options

Use the `default:` section to specify site-wide options that *can* be overridden by individual users. Multiple options may be specified on a single line if separated by commas.

In the example below, the `mode` option is set to `text` default. Because this option is set in the `default:` section, it can be easily overridden by individual users.

```
default:
mode=text
```

Note

The shell script used to run the Co:Z SFTP subsystem (`sftp-server.sh`) and the sample `cozsftp` batch scripts (`sftp_connect.sh, sftp_get.sh`, etc.) export LC_ALL=C to ensure proper shell script and C runtime execution. As a result, the default codepage for file transfer will be set to the z/OS platform default of `IBM-1047`. In order to set Locale specific codepage defaults, the `default:` section of the configuration files can be used:

```
# cozsftp_config (client)
# Set the default codepage for file transfers to EBCDIC Finnish/Swedish
default:
clientcp=IBM-1138
```

```
# cozsftp_server_config (server)
# Sets the default codepage for file transfers to EBCDIC Finnish/Swedish
default:
servercp=IBM-1138
```

C.4 Specifying file pattern specific options

It is often useful to have a set of custom options associated with specific files and/or datasets. For example, transferring all files with the `.pax` extension in binary mode. The `pattern` sections of the configuration files enable file and dataset names matching a specific POSIX *glob pattern* to automatically have specific options applied regardless of the options currently in place.

`pattern` sections can be supplied in the site (`/etc/ssh`) versions of the config files and may also be supplied in copies of these files located in the user's `$HOME/.ssh` directory.

Note

In some cases, it may not be possible for this file to be located in `$HOME/.ssh`; if this is the case, this location can be overridden during Co:Z SFTP server or cozsftp startup. For details, see *the section called “Sitewide server configuration”* and *the section called “Sitewide client configuration”*.

Note

A specific pattern may only be defined once; subsequent definitions read from the config file(s) are ignored.

When a `put` or `get` command is issued, the file or dataset name is checked against the patterns in the order that they were originally read. The options associated with the *first* matching pattern (if any) are applied to that specific file transfer. If an option is not defined by the pattern, it is left unchanged. Once the transfer completes, the overridden options are restored.

Pattern sections have the following syntax:

```
pattern: [//]<glob_pattern>
pattern-get: [//]<glob_pattern>
pattern-put: [//]<glob_pattern>
```

If double slashes (//) precede the pattern, it is used to match dataset names, otherwise it is used to match POSIX pathnames. Matching is performed on the name after it has been normalized (e.g. embedded slashes in a dataset name are converted to periods and the characters are converted to uppercase). Please note that DD names will not be **not** resolved to their catalog name prior to matching.

Patterns follow the UNIX *glob pattern* syntax, where the wildcard characters `?` (match exactly one character) and `*` (match zero or more characters) can be used in conjunction with literal characters to provide a match pattern. For a complete description of the pattern syntax, see the “File name generation” section of the **sh** command documentation in the *z/OS Unix System Services Command Reference*.

Patterns in either the `pattern:` or `pattern-get:` sections are used to match files that are involved in SFTP **get** operations. Patterns in the `pattern:` or `pattern-put:` sections are used to match files that are involved in SFTP **put** operations. The same pattern may be defined in both a `pattern-get:` and `pattern-put:` section (with potentially different transfer options). A pattern specified in a `pattern:` section will apply to both operations. **Note:** If a pattern is defined in a `pattern:` section, it may **not** also be defined in a `pattern-get:` / `pattern-put:` section, and vice versa.

Determining which argument of the **get** or **put** command is used to match a pattern depends on which Co:Z component is being used:

sftp-server patterns (defined in the `cozsftp_server_config` files) are applied as follows:

```
sftp> get file-or-dsn <dest>    # pattern-get: or pattern: section
sftp> put <source> file-or-dsn  # pattern-put: or pattern: section
```

cozsftp patterns (defined in the `cozsftp_config` files) are applied as follows:

```
cozsftp> get <source> file-or-dsn  # pattern-get: or pattern: section
cozsftp> put file-or-dsn <dest>    # pattern-put: or pattern: section
```

Pattern examples

Setting text mode transfer for all members of a PDS

In the following example, a user specifies in $HOME/.ssh/cozsftp_config:

```
pattern: //*.JCL(*)
```

```
mode=text
```

And in a **cozsftp** session issues the following:

```
$ cozsftp user@host
Co:Z SFTP version: 1.9.3 (5.0p1) 2011-09-01
Copyright (C) Dovetailed Technologies, LLC. 2011. All rights reserved.
Connecting to host...
user@host's password: *****
cozsftp>lzopts mode=binary
cozsftp>get myjcl //HLQ.DEV.JCL(FOO)
```

Because the target name matches the pattern, the file `myjcl` will be transferred as in text mode even though the current mode setting is binary.

Automatically set dataset allocation parameters

In the following example, consider the Co:Z SFTP server configuration file /etc/ssh/cozsftp_server_config:

```
pattern: //*.PARTNER.TRANS*
space=cyl.3.2,recfm=fb,lrecl=80
```

And a remote sftp session issues the following:

```
sftp> put trans0923 //HLQ.PARTNER.TRANS0923
```

Assuming the dataset `HLQ.PARTNER.TRANS0923` doesn't already exist, a new dataset with that name will be allocated with allocation parameters associated with the pattern. This example shows how a server can be setup to automatically allocate incoming datasets based on a predefined name pattern.

Pattern selection determined by first match

For the examples that follow, consider the following configuration files excerpts:

```
(from $HOME/.ssh/cozsftp_server_config)
pattern: *.txt
mode=text,clientcp=1252,linerule=crlf
```

```
(from /etc/ssh/cozsftp_server_config)
pattern: *.zip
mode=binary
pattern: *.pax
mode=binary
pattern: *.txt
mode=text,linerule=lf
```

```
sftp> get myarchive.pax     ❶
sftp> get mynotes.txt     ❷
```

❶ The file will be transferred in binary mode because it matches the site specified pattern (via `/etc/ssh/cozsftp_server_config`) for files with a `.pax` extension.

❷ The file myfile.txt will be transferred in text mode with a client code page of 1252 and a linerule of `crlf`. While the `.txt` extension could match two of the specified patterns, the one processed first (via `$HOME/.ssh/cozsftp_server_config`) is selected. This is an example of how an individual user can override site behavior for a specific need (e.g. a Windows client platform).

Appendix D. Dataset Name Determination

When issuing a `put` command to create a dataset, or `get` to a local dataset using the cozsftp client, the resulting dataset name is determined as follows:

```
put myfile //HLQ.LEVEL  (a remote client using the Co:Z SFTP server)
get myfile //HLQ.LEVEL  (using the cozsftp client)
```

Table D.1. Dataset Name determination

Case	Condition	Dataset Name	Notes
1	HLQ.LEVEL is a Sequential Dataset	HLQ.LEVEL	Replaces existing SEQ dataset
2	HLQ.LEVEL is a PDS	HLQ.LEVEL(MYFILE)	Creates or replaces member named `MYFILE` in PDS
3	HLQ.LEVEL is not a dataset, but HLQ.LEVEL.XXX names exist in catalog	HLQ.LEVEL.MYFILE	Create or replace SEQ dataset
4	HLQ.LEVEL is not a dataset, and no HLQ.LEVEL.XXX names exist in catalog	HLQ.LEVEL	Creates new SEQ dataset

In most cases, this is acceptable behavior. However, there are cases where the supplied name should be treated as a dataset rather than a "directory" (as in Case 3 above). If this is the required behavior, a different dataset prefix can be supplied: `//!` or `/-/!`.

```
put myfile //!HLQ.LEVEL  (a remote client using the Co:Z SFTP server)
get myfile //!HLQ.LEVEL  (using the cozsftp client)
```

Note: In release 2.4.0, this was relaxed so that the `'!'` decorator may appear anywhere in the data set name.

*Table D.2. Dataset Name determination - **DSN contains "!" decorator***

Case	Condition	Dataset Name	Notes
1	HLQ.LEVEL is a Sequential Dataset	HLQ.LEVEL	Replaces existing SEQ dataset

Case	Condition	Dataset Name	Notes
2	HLQ.LEVEL is a PDS	HLQ.LEVEL(MYFILE)	Creates or replaces member named `MYFILE` in PDS
3	HLQ.LEVEL is not a dataset	HLQ.LEVEL	Creates new SEQ dataset

D.1 maxdsndirlevels option

In *release 2.4.0*, the `maxdsndirlevels` setting was added to specify the maximum number of levels that a data set name can have before it is always considered as a new file rather than a (pseudo) directory.

For example:

```
(a remote client connected to Co:Z SFTP server)
sftp> ls -alf //kirk.dsn.test
Volume   Referred  Ext  Tracks    Used Recfm Lrecl BlkSz Dsorg  Dsname
VPWRKC  2013/06/06   1       1       1  U        0  6144  PS    KIRK.DSN.TEST.TST1
VPWRKB  2013/06/06   1      15       1  FB      80 27920  PS    KIRK.DSN.TEST.TXT2
sftp> put local.file //kirk.dsn.test
Uploading local.file to //kirk.dsn.test/local.file ❶
...
sftp> put local.file //!kirk.dsn.test  ❷
Uploading local.file to //kirk.dsn.test
...
sftp> rm //kirk.dsn.test
sftp> ls /+maxdsndirlevels=2  ❸
/+maxdsndirlevels=2
sftp> put local.file //kirk.dsn.test  ❹
Uploading local.file to //kirk.dsn.test
...
sftp> rm //kirk.dsn.test
sftp> cd //kirk.dsn.test  ❺
Couldn't stat remote file: No such file or directory
```

❶ //kirk.dsn.test is treated as a "directory", since there is no data set with that name but there are data sets at lower levels. Since the sftp client sees a directory, it will create a new file name in that directory. The resulting DSN is `KIRK.DSN.TEST.LOCAL.FILE`

❷ The use of the `'!'` character in the DSN will force Co:Z SFTP server to tell the client that it is a non-existant file, rather than a directory. The resulting DSN is: `KIRK.DSN.TEST`

❸ Setting this option will mean that DSNs with more than two levels are never considered as pseudo directories.

❹ The use of the `'!'` decorator is not required. The resulting DSN is: `KIRK.DSN.TEST`

❺ Since this DSN has more levels than `maxdsndirlevels`, you can not "change directory" to it.

Recommendation: When creating new data sets where it is possible that data sets exist at lower levels, use the `//!` or `/-/!` syntax or the `maxdsndirlevels` option.

Appendix E. SMF Information

E.1 IBM FTP-compatible SMF 119 record subtypes

Co:Z SFTP supports recording SMF type 119 records that are compatible with the following IBM FTP records:

- *Subtype 3 - FTP client transfer completion*
- *Subtype 70 - FTP server transfer completion*
- *Subtype 100 - FTP server transfer initialization* (real-time SMF data NMI only)
- *Subtype 101 - FTP client transfer initialization* (real-time SMF data NMI only)

Refer to the *z/OS Communications Server: IP Programmer's Guide and Reference* for complete documentation on FTP SMF type 119 records. Section *SMF Record Formats* below highlights Co:Z SFTP specific field information.

E.2 New SMF 119 record subtypes

In addition to standard FTP completion/initialization records above, Co:Z SFTP also creates the following SMF 119 record subtypes:

- *Subtype 192 - Co:Z SFTP server log messages*
- *Subtype 193 - Co:Z SFTP client log messages*
- *Subtype 194 - Co:Z SFTP server interim transfer* (real-time SMF data NMI only)
- *Subtype 195 - Co:Z SFTP client interim transfer* (real-time SMF data NMI only)

For more information on the Co:Z SFTP specific type 119 records, see section *SMF Record Formats*.

Note: Record types 100, 101, 194, and 195 are never written as real SMF records

E.3 Enabling SMF recording

In order to enable recording of Co:Z SFTP SMF 119 records, you must:

1. configure SMF to allow recording these records and subtypes. See *z/OS MVS System Management Facilities (SMF)* for more information.
2. permit the users running Co:Z SFTP client or server jobs `READ` access to the `BPX.SMF FACILITY` class resource. Alternatively, you may also use type/subtype specific permissions (see next section).
3. the `nosmf` configuration option must not be set. See *http://dovetail.com/docs/sftp/options.html#options_misc* for more information.
4. in order to get accurate local and remote host/port information for client SMF records, the program `COZ_HOME/bin/ssh-socket-info` is called by Co:Z once the child ssh session is established.

This program uses the IBM EZBNMIFR network management API, which requires the ssh-socket-info program to be APF authorized. The Co:Z installer will attempt to set the "+a" extattr bit on this program, but will only succeed if the installing userid has READ access to the BPX.FILEATTR.APF SAF resource. If for some reason, this program is not APF authorized, Co:Z SFTP will operate properly, but the SMF socket information will not be accurate in client SMF records.

Using SMF type/subtype specific permissions

Introduced by *APAR OA48775*, z/OS now allows non-authorized programs to write specific SMF record types/subtypes. This is supported starting with Co:Z SFTP 4.5.0 using the following steps:

- Permit the users running Co:Z SFTP client or server jobs `READ` access to `BPX.SMF.119.n` resource, for n = {3, 70, 192, 193}.
- The Co:Z SFTP client and server programs must be **program controlled**. Starting in release 4.5.0, The Co:Z installer will attempt to set the "+p" extattr bit on the Co:Z SFTP client and server programs (`cozsftp_cmd` and `sftp-server`) in the install directory.
- For running the Co:Z SFTP client in batch, you must explicitly mark the `COZ.LOADLIB` dataset as program controlled. If you are using Co:Z SFTP server user exits, this load library must also be marked as program controlled.
- The address spaces where you run Co:Z SFTP must remain program-controlled "clean" - in other words, you may not run any non-program controlled commands in the same address space prior to running Co:Z SFTP:
  - For Co:Z SFTP server, do not run any non-program controlled commands in your system or user-level `sftp-server.rc` scripts. Commands may be run using `$(cmd ...)` or `` `cmd ...` `` or by temorarily using `export _BPX_SHAREAS=NO` and back to `YES` around the command, since these will not run in the same address space.
  - For Co:Z SFTP client, watch for commands that might run in the script that you use to invoke the `cozsftp` command, or in the `/etc/profile` or `$HOME/.profile` scripts. Starting in 4.5.0, the sample `SFTPPROC` will start the z/OS shell in the same address space but with `_BPX_SHAREAS=NO`. Any commands issued by the profile scripts prior setting `_BPX_SHAREAS=YES` will run in a separate address space to avoid dirtying the program-controlled environment.

    To diagnose program control issues in client batch jobs, run the step with: `ARGS='-LD /bin/sh -Lx'` to enable COZBATCH and z/OS shell tracing.

E.4 Using the Real-Time Co:Z SMF Interface

The Co:Z SFTP client and server will also write SMF 119 records to a Unix datagram socket if it is available. By default, the name of the socket is `/var/log/cozsftp.smf.sock` unless overridden by the `SFTP_SMF_SOCK` environment variable. This interface is useful in managed file transfer environments that need real-time access to file transfer events. The real-time interface is independent of actual SMF recording - you may use either real SMF recording, the datagram socket, or both.

SMF 119 record subtypes related to interim file transfer logging are only written using the real-time Co:Z SMF Interface. Real-time logging of these records is enabled by setting the option `intermlogging=nnnn` where nnnn

is the interval in seconds. Interim log messages are written during a file transfer. When this feature is enabled and a file transfer is initiated, an initialization record is written at the start of the transfer (*subtype 100* by the Co:Z SFTP server, *subtype 101* by the Co:Z SFTP client). At the interval specified, interim records (*subtype 194* by the Co:Z SFTP server, *subtype 195* by the Co:Z SFTP client) are logged capturing the bytes transferred at the time identified in the record header. See *Miscellaneous options*.

To use this facility, you must write a program that creates this Unix-domain socket and receives datagram messages from it. Each message will be a SMF record image from a Co:Z SFTP client or server running on the same system. A sample C++ program, `CoZSmfServer.C`, demonstrates how to use this facility. See the documentation and build instructions in `$COZ_HOME/samples/smfapi/CoZSmfServer.C`. This sample illustrates the following scenarios: consolidation of BPX.SMF authorization to a single job or user, passing of SMF records in real-time to another program, and real-time logging of initialization, interim and completion file transfer SMF records.

E.5 SMF Record Formats

The *z/OS Communications Server: IP Programmer's Guide and Reference* contains complete documentation on FTP SMF type 119 records. This section highlights Co:Z SFTP specific field information (shown in **bold**) as well as record formats for Co:Z SFTP type 119 subtypes.

Common Sections

- TCP/IP identification

Offset	Length	Format	Description
0	8	EBCIDIC	System name
8	8	EBCIDIC	Sysplex name
16	8	EBCIDIC	TCP/IP stack name
24	8	EBCIDIC	TCP/IP release identifier. **Set to '011100' for V1 Release 11.**
32	8	EBCDIC	TCP/IP subcomponent. **Set to 'SFTPS' (SFTP server) or 'SFTPC' (SFTP client).**
40	8	EBCDIC	ASName
48	8	EBCDIC	UserID
56	4	binary	ASID
60	1	binary	Reason. **Set to X'08', Event SMF record.**
61	3	binary	reserved

- FTP security

Offset	Length	Format	Description
0	1	EBCIDIC	Protection Mechanism. **Set to T: TLS.**

Offset	Length	Format	Description
1	1	EBCIDIC	Control Connection Protection Level. **Set to P: Private.**
2	1	EBCDIC	Data Connection Protection Level. **Set to P: Private.**
3	1	EBCDIC	Login Method. **Set to P: Password.**
4	8	EBCDIC	Protocol level. **Set to blanks.**
12	20	EBCDIC	Cipher Specification. **Set to blanks.**
32	4	EBCDIC	Protection buffer size. **Set to 0.**
36	2	binary	Reserved

Subtype 3 - FTP client transfer completion

- Self defining section

  The self-defining section identifies 6 triplets, although 7 are allocated. The triplets are:

  - TCP/IP identification
  - FTP client transfer completion
  - FTP client transfer completion associated data set name
  - FTP client SOCKS - **triplet set to zero**
  - FTP security
  - FTP user name

- FTP client transfer completion

  Several fields noted below are set from ssh socket information, if available. See section *Enabling SMF recording* for additional information.

Offset	Length	Format	Description
0	4	EBCIDIC	FTP command
4	4	EBCIDIC	Local file type
8	16	binary	Remote IP address (data connection). **Set from ssh socket information, if available.**
24	16	binary	Local IP address (data connection). **Set from ssh socket information, if available.**
40	2	binary	Local port (data connection). **Set from ssh socket**

Offset	Length	Format	Description
			information, if available.
42	2	binary	Remote port (data connection). **Set from ssh socket information, if available.**
44	16	binary	Remote IP address (control connection). **Set equal to the data connection value.**
60	16	binary	Local IP address (control connection). **Set equal to the data connection value.**
76	2	binary	Remote port (control connection). **Set equal to the data connection value.**
78	2	binary	Local port (control connection). **Set equal to the data connection value.**
80	8	EBCIDIC	Server user id
88	8	EBCIDIC	Local user id
96	1	EBCIDIC	Data format
97	1	EBCIDIC	Transfer mode
98	1	EBCIDIC	Structure
99	1	EBCIDIC	Data set type
100	4	binary	Transfer start time
104	4	packed	Transfer start date
108	4	binary	Transfer end time
112	4	packed	Transfer end date
116	4	binary	Transfer duration
120	8	binary	Transmission byte count
128	4	EBCIDIC	Last server reply
132	8	EBCIDIC	PDS member name
140	8	EBCIDIC	Host name
148	8	EBCIDIC	Abnormal end information
156	8	floating point hex	Transmission byte count (float)
164	4	binary	TCP connection ID (control connection). **Set from ssh socket information, if available.**
168	4	binary	TCP connection ID (data connection). **Set equal to**

Offset	Length	Format	Description
			the control connection value.

Subtype 70 - FTP server transfer completion

- Self defining section

  The self-defining section identifies 6 triplets, although 7 are allocated. The triplets are:

  - TCP/IP identification
  - FTP server transfer completion
  - FTP server host name
  - FTP server first associated data set name
  - FTP server second associated data set name
  - FTP security
- FTP server transfer completion

Offset	Length	Format	Description
0	1	binary	FTP operation
1	3	binary	reserved
4	4	EBCIDIC	FTP command
8	4	EBCIDIC	Local file type
12	16	binary	Remote IP address (data connection)
28	16	binary	Local IP address (data connection)
44	2	binary	Local port (data connection)
46	2	binary	Remote port (data connection)
48	16	binary	Remote IP address (control connection). **Set equal to the data connection value.**
64	16	binary	Local IP address (control connection). **Set equal to the data connection value.**
80	2	binary	Remote port (control connection). **Set equal to the data connection value.**
82	2	binary	Local port (control connection). **Set equal to the data connection value.**

Offset	Length	Format	Description
84	8	EBCIDIC	Client user id on server
92	1	EBCIDIC	Data type
93	1	EBCIDIC	Transmission mode
94	1	EBCIDIC	Data Structure
95	1	EBCIDIC	Data set type
96	4	binary	Transfer start time
100	4	packed	Transfer start date
104	4	binary	Transfer end time
108	4	packed	Transfer end date
112	4	binary	Transfer duration
116	8	binary	Transmission byte count
124	4	EBCIDIC	Last reply to client
128	8	EBCIDIC	PDS member name
136	8	EBCIDIC	Abnormal end information
144	8	EBCIDIC	Second PDS member name
152	8	floating point hex	Transmission byte count (float)
160	4	binary	TCP connection ID (control connection). **Set to 0.**
164	4	binary	TCP connection ID (data connection). **Set to 0.**
168	15	EBCIDIC	Session id. **Set to a generated value: jobname followed by the last five digits of the process id.**
183	1	binary	reserved

Subtype 100 - FTP server transfer initialization (real-time SMF data NMI record format)

Real-time transfer SMF records are not written by default. Refer to *Miscellaneous options* for information on setting the `interimlogging` option to enable this feature. Additionly, see *Using the real-time Co:Z SMF interface* for information on accessing real-time SMF records.

- Self defining section

  The self-defining section identifies 6 triplets, although 7 are allocated. The triplets are:

  - TCP/IP identification
  - FTP server transfer initialization
  - FTP server host name
  - FTP server first associated data set name
  - FTP server second associated data set name
  - FTP security
- FTP server transfer initialization

Offset	Length	Format	Description
0	1	binary	FTP operation
1	1	binary	Passive or active mode data connection. **Set to X'00': Active using default IP and port.**
2	2	binary	reserved
4	4	EBCIDIC	FTP command
8	4	EBCIDIC	Local file type
12	16	binary	Remote IP address (data connection)
28	16	binary	Local IP address (data connection)
44	2	binary	Local port (data connection)
46	2	binary	Remote port (data connection)
48	16	binary	Remote IP address (control connection). **Set equal to the data connection value.**
64	16	binary	Local IP address (control connection). **Set equal to the data connection value.**
80	2	binary	Remote port (control connection). **Set equal to the data connection value.**
82	2	binary	Local port (control connection). **Set equal to the data connection value.**
84	8	EBCIDIC	Client user id on server
92	1	EBCIDIC	Data type
93	1	EBCIDIC	Transmission mode
94	1	EBCIDIC	Data Structure
95	1	EBCIDIC	Data set type

Offset	Length	Format	Description
96	4	binary	Data connection start time. **Set to the start time of the session.**
100	4	packed	Data connection start date. **Set to the start date of the session.**
104	4	binary	Control connection start time. **Set equal to the data connection value.**
108	4	packed	Control connection start date. **Set equal to the data connection value.**
112	8	EBCIDIC	PDS member name
120	8	EBCIDIC	Second PDS member name
128	4	binary	TCP connection ID (control connection). **Set to 0.**
132	4	binary	TCP connection ID (data connection). **Set to 0.**
136	15	EBCIDIC	Session id. **Set to a generated value: jobname followed by the last five digits of the process id.**
151	1	binary	reserved

Subtype 101 - FTP client transfer initialization (real-time SMF data NMI record format)

Real-time transfer SMF records are not written by default. Refer to *Miscellaneous options* for information on setting the `interimlogging` option to enable this feature. Additionally, see *Using the real-time Co:Z SMF interface* for information on accessing real-time SMF records.

- Self defining section

  The self-defining section identifies 6 triplets, although 7 are allocated. The triplets are:

  - TCP/IP identification
  - FTP client transfer initialization
  - FTP client associated data set name
  - FTP client SOCKS - **triplet set to zero**
  - FTP security
  - FTP user name
- FTP client transfer initialization

Several fields noted below are set from ssh socket information, if available. See section *Enabling SMF recording* for additional information.

Offset	Length	Format	Description
0	4	EBCIDIC	FTP command
4	4	EBCIDIC	Local file type
8	16	binary	Remote IP address (data connection). **Set from ssh socket information, if available.**
24	16	binary	Local IP address (data connection)**Set from ssh socket information, if available.**
40	2	binary	Local port (data connection)**Set from ssh socket information, if available.**
42	2	binary	Remote port (data connection)**Set from ssh socket information, if available.**
44	16	binary	Remote IP address (control connection). **Set equal to the data connection value.**
60	16	binary	Local IP address (control connection). **Set equal to the data connection value.**
76	2	binary	Remote port (control connection). **Set equal to the data connection value.**
78	2	binary	Local port (control connection). **Set equal to the data connection value.**
80	8	EBCIDIC	Server user id
88	8	EBCIDIC	Local user id
96	1	EBCIDIC	Data format
97	1	EBCIDIC	Transfer mode
98	1	EBCIDIC	Structure
99	1	EBCIDIC	Data set type
100	4	binary	Start time of data connection. **Set to the start time of the session.**
104	4	packed	Start date of data connection. **Set to the start date of the session.**
108	4	binary	Start time of control connection. **Set equal to the data connection value.**
112	4	packed	Start date of control connection. **Set equal to the data connection value.**

Offset	Length	Format	Description
116	8	EBCIDIC	PDS member name
124	1	EBCIDIC	Passive or active mode data connection. **Set to X'00': Active using default IP and port.**
125	3	binary	reserved
128	4	binary	TCP connection ID (control connection). **Set from ssh socket information, if available.**
132	4	binary	TCP connection ID (data connection). **Set equal to the control connection value.**

Subtype 192 - Co:Z SFTP server log messages

- Self defining section

  The self-defining section identifies 3 triplets, although 7 are allocated. The triplets are:

  - TCP/IP identification
  - Socket connection
  - Co:Z SFTP messages

- Socket connection

Offset	Length	Format	Description
0	16	binary	Remote IP address
16	16	binary	Local IP address
32	2	binary	Remote port number
34	2	binary	Local port number
36	15	EBCDIC	FTP session ID. **Set to a generated value: jobname followed by at most the last five digits of the process id.**
51	1	binary	reserved

- Co:Z SFTP messages

  This section contains Co:Z SFTP messages, informational level or above, that were associated with the previous transfer. One or more message sub-sections may be included, each with the following layout:

Offset	Length	Format	Description
0	4	binary	Time (in local time)

Offset	Length	Format	Description
4	4	Packed	Date (in local time)
8	2	binary	Length of message that follows
10	variable	EBCDIC	Message text

Subtype 193 - Co:Z SFTP client log messages

- Self defining section

  The self-defining section identifies 3 triplets, although 7 are allocated. The triplets are:

  - TCP/IP identification
  - Socket connection
  - Co:Z SFTP messages

- Socket connection

  Fields noted below are set from ssh socket information, if available. See section *Enabling SMF recording* for additional information.

Offset	Length	Format	Description
0	16	binary	Remote IP address. **Set from ssh socket information, if available.**
16	16	binary	Local IP address. **Set from ssh socket information, if available.**
32	2	binary	Remote port number. **Set from ssh socket information, if available.**
34	2	binary	Local port number. **Set from ssh socket information, if available.**
36	15	EBCDIC	FTP session ID. **Set to blank.**
51	1	binary	reserved

- Co:Z SFTP messages

  This section contains Co:Z SFTP messages, informational level or above, that were associated with the previous transfer. One or more message sub-sections may be included, each with the following layout:

Offset	Length	Format	Description
0	4	binary	Time (in local time)

Offset	Length	Format	Description
4	4	Packed	Date (in local time)
8	2	binary	Length of message that follows
10	variable	EBCDIC	Message text

Subtype 194 - Co:Z SFTP server interim transfer (real-time Co:Z SMF interface)

Real-time transfer SMF records are not written by default. Refer to *Miscellaneous options* for information on setting the `interimlogging` option to enable this feature. Additionally, see *Using the real-time Co:Z SMF interface* for information on accessing real-time SMF records.

- Self defining section

  The self-defining section identifies 7 triplets. The triplets are:

  - TCP/IP identification
  - FTP server transfer initialization - **Set equal to FTP server transfer initialization (subtype 100)**
  - FTP server host name
  - FTP server first associated data set name
  - FTP server second associated data set name
  - FTP security
  - FTP interim transfer
- FTP interim transfer section

Offset	Length	Format	Description
0	8	binary	Estimated file size (bytes). Set to -1 on put (write) or if read and source file size is unknown.
8	8	binary	Estimated file size (bytes float). Set to -1 on put (write) or if read and source file size is unknown.
16	8	binary	Interim transmission byte count
24	8	binary	Interim transmission byte count (float)

Subtype 195 - Co:Z SFTP client interim transfer (real-time Co:Z SMF interface)

Real-time transfer SMF records are not written by default. Refer to *Miscellaneous options* for information on setting the `interimlogging` option to enable this feature. Additionally, see *Using the real-time Co:Z SMF interface* for information on accessing real-time SMF records.

- Self defining section

  The self-defining section identifies 7 triplets. The triplets are:

  - TCP/IP identification
  - FTP client transfer initialization - **Set equal to FTP client transfer initialization (subtype 101)**
  - FTP client associated data set name
  - FTP client SOCKS - **triplet set to zero**
  - FTP security
  - FTP user name
  - FTP interim transfer

- FTP interim transfer section

Offset	Length	Format	Description
0	8	binary	Estimated file size (bytes). Set to -1 on get (write) or if read and source file size is unknown.
8	8	binary	Estimated file size (bytes float). Set to -1 on get (write) or if read and source file size is unknown.
16	8	binary	Interim transmission byte count
24	8	binary	Interim transmission byte count (float)

Appendix F. Client Authentication Mechanisms

Running the Co:Z SFTP client and/or the Co:Z Launcher requires that the z/OS ssh client can authenticate with the Target System ssh server. Several authentication choices are available from z/OS; site policies will usually dictate which is best.

One of the following authentication mechanisms should be performed on z/OS from **each** userid that will be used to execute the Co:Z SFTP or Co:Z Launcher jobs.

- Interactive password: *Section F.1, "Interactive password authentication"*. **Note:** this mechanism requires user keyboard interaction, so it will not work in batch. It should only be used for command line invocations of the Co:Z SFTP client.
- OpenSSH ASK_PASS (read a password from a dataset): *Section F.3, "OpenSSH SSH_ASKPASS authentication"*.
- Conventional OpenSSH keypairs: *Section F.2, "OpenSSH keypair authentication"*.
- RACF Digital Certificates: *Section F.4, "RACF Digital Certificate authentication"*.

F.1 Interactive password authentication

This is the simplest form of OpenSSH client authentication and requires no additional setup. It can only be used from a terminal (Unix TTY) connected shell where the user can supply the target system password. Due to this requirement, it is not suitable for z/OS batch programs and is therefore not an option for running the Co:Z Launcher or batch Co:Z SFTP. It *is* suitable for interactive shell invocations of Co:Z SFTP.

Note: The IBM Ported Tools OpenSSH client v1.2 will not run from a TSO OMVS shell session, so if you want to interactively use the Co:Z SFTP client you must use a z/OS shell under telnet, rlogin, or ssh. Later IBM Ported Tools OpenSSH and z/OS OpenSSH client versions are supported from a TSO OMVS shell session, but do not allow a password to be entered from a 3270 terminal.

F.2 OpenSSH keypair authentication

This is the conventional mechanism for performing OpenSSH client authentication. A public/private key pair is generated on z/OS. The private key is kept (protected) in the user's `~/.ssh` directory. The public key is stored on each target system in the user's `~/.ssh/authorized_keys` file. The following steps describe how to generate and use an OpenSSH keypair:

Note: Proceed with caution if you have more than one userid mapped to the same `uid` number (an unfortunately common occurrence on z/OS USS). The default key storage home directory is hard to predict.

1. Generate a keypair using `ssh-keygen`:

```
$ mkdir ~/.ssh
$ chmod 700 ~/.ssh
```

```
$ ssh-keygen -t rsa -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (/home/<userid>/.ssh/id_rsa): <enter>
Enter passphrase (empty for no passphrase): <enter>
Enter same passphrase again: <enter>
Your identification has been saved in /home/<userid>/.ssh/id_rsa.
Your public key has been saved in /home/<userid>/.ssh/id_rsa.pub.
The key fingerprint is:
dd:ff:00:87:43:11:fa:7b:0d:84:3a:19:3b:7f:5d:2e <userid>@<host>
The key's randomart image is:
+--[ RSA 2048]----+
|oEoo .           |
|o.. + o .        |
|.  ..= o .       |
|.   .o=o.        |
|.   ...o+.       |
| . .   ...       |
|  o     o        |
| o     o         |
|  .              |
+-----------------+
```

The private key file `id_rsa` will be generated without a passphrase so that Co:Z can run in batch. It is therefore important that this file is protected with file permissions and/or ACLs that only allow the owning userid to read the file.

2. Move a copy of the public key to the target system:

```
ZOS$ sftp -oPort=<port>  cozuser@linux1.myco.com
Connecting to n.n.n.n...
cozuser@linux1.myco.com's password:  ******
sftp>  ascii
Sets the file transfer type to ASCII.
sftp>  cd .ssh
sftp>  put -P id_rsa.pub authorized_keys
Uploading id_rsa.pub to /home/sgoetze/.ssh/authorized_keys
id_rsa.pub                                    100%  601     0.6KB/s   00:00
sftp> quit
```

Note: If you are adding more than one public key to `authorized_keys`, then you must log in to the remote system and append the new public key line to authorized_keys. Be careful that you don't replace an existing authorized_keys file.

Note: The `authorized_keys` file, the `.ssh` directory, and the home directory must not be writable by any user other than the owning userid. For details on required file permissions, see the section *OpenSSH files Quick Reference / User-generated files* in *z/OS OpenSSH User's Guide*.

Note: For more information on using SSH key authentication, see our webinar archives: *IBM Ported Tools for z/OS: OpenSSH - Key Authentication*.

F.3 OpenSSH SSH_ASKPASS authentication

OpenSSH supports the use of the `SSH_ASKPASS` environment variable to point to a program that will read a password, without keyboard interaction.

Using `SSH_ASKPASS` with OpenSSH requires that other ssh settings and environment variables are configured. The `SFTPSAMP` and `RUNSFTP` sample JCL members illustrate how to do this with Co:Z SFTP; the `RUNLNCHP` sample JCL shows how for Co:Z Launcher. With these samples, a dataset must be created (e.g.)`//HLQ.PASSWD(SITE1)` that contains a single line with the password starting in the first column and *without* line numbers. The dataset should be protected with RACF so that it cannot be read except by the required jobs.

F.4 RACF Digital Certificate authentication

Traditional OpenSSH keypairs and SSH_ASKPASS are convenient, but some sites have strict policies about keeping key material in RACF (or another security package), or even ICSF hardware. The z/OS Communcations Server FTP command can exploit SAF/RACF Digital Certificates for authentication and encryption. z/OS OpenSSH allows you to use keys that are stored in SAF/RACF certificates. The Co:Z toolkit provides a similar capability via its `saf-ssh-agent`, but also allows you to use certificates with private keys stored in ICSF managed hardware.

An existing SAF/RACF key ring and client certificate set up for use with the z/OS FTP client may be used with Co:Z Launcher and the Co:Z SFTP client.

The following steps describe how to create an RSA RACF Digital Certificate, export its public key in OpenSSH compatible format, and transfer the public key to the target system.

1. Create a Key Ring and RSA Digital Certificate:

   Note: In order to create RACF Digital Certificates, certain RACF permissions must be held. This step is typically performed by an administrator; the permissions required are *not* required for the user to access the certificate (see below). For details, see the chapter *RACF and Digital Certificates z/OS Security Server RACF Security Administrator's Guide (SA22-7683)*.

   This JCL is located in RACDCERT member of the COZ.SAMPJCL PDS. It will create an RSA Digital Certificate labeled `MY-CERT` held in the key ring `MY-RING`.

   It is also possible to skip creating a key ring - any certificate automatically belongs to the user's *virtual key ring*, and may be referenced by using the special key ring name "*". For more information on using SAF/RACF key rings with OpenSSH, see our webinar archives: *IBM Ported Tools for z/OS: OpenSSH - Using Key Rings*.

```
//COZUSERJ  JOB (),'',MSGCLASS=H,NOTIFY=&SYSUID
//*
//  EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
```

```
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *

 /* Generate a self-signed RSA certificate to use          */
 /* for SSH client authentication.                         */
 /* A certificate signed by your CA will also work.        */
 RACDCERT ID(COZUSER) GENCERT                              + ❶
            SUBJECTSDN(                                    +
               CN('First Lastname' )                       + ❷
                O('My Company')                            + ❷
               OU('Development')                           + ❷
                C('US')                                    + ❷
            )                                              + ❷
            NOTAFTER(DATE(2016-12-31))                     + ❸
            WITHLABEL('MY-CERT')                           +
            ICSF                                             ❹

 /* Create a KEYRING for the user (skip for virtual keyring) */
 RACDCERT ID(COZUSER) ADDRING(MY-RING)                       ❶

 /* Connect the certificate to the ring (skip for virtual)   */
 RACDCERT ID(COZUSER) CONNECT (                            + ❶
                          ID(COZUSER)                      + ❶
                          LABEL('MY-CERT')                 +
                          RING(MY-RING)                    +
                          DEFAULT                          + ❺
                          USAGE(PERSONAL) )
 /* Refresh if RACLISTed                                   */
 SETROPTS RACLIST(DIGTCERT, DIGTRING) REFRESH

 /* List the user's certs                                  */
 RACDCERT ID(COZUSER) LIST                                   ❶
//
```

- ❶ Change the string `COZUSER` to the MVS userid that will own and use the certificate.
- ❷ Change the subject DSN fields according to your company's standards.
- ❸ Specifies the expiry date of the certificate, otherwise it defaults to one year.
- ❹ Optional keywords `ICSF` or `PCICC`, may be specified. When not specified, the generated certificate is stored in the RACF database as a non-ICSF RSA key. When one of `ICSF` or `PCICC` is specified, the certificate generated is stored in the ICSF PKA key data set. The `ICSF` and `PCICC` keywords require ICSF to be started as well as CSFKEYS authorities. For more information, see: "z/OS ICSF Administrator's Guide SA22-7251" - "Using RACF to protect Keys and Services".

  Note: If using `ICSF` or `PCICC`, you will only be able to use the Co:Z `saf-ssh-agent`, and not the IdentityKeyRingLabel support in z/OS OpenSSH.
- ❺ Makes this certificate the default in the ring. This allows the user to specify just the key ring name in order to access the certificate.

2. Export an OpenSSH version of the certificate's public key:

Note: This and the remaining steps are performed by the user. In order to access the key ring and certificate, the user must have the following SAF/RACF permissions:

- `CLASS(FACILITY) IRR.DIGTCERT.LISTRING ACCESS(READ)`
- `CLASS(CSFSERV) CSFDSG ACCESS(READ)`
- `CLASS(CSFSERV) CSFDSV ACCESS(READ)`

Public key extraction is performed using Co:Z's `saf-ssh-agent` and the `-x` option. If the `-f` option is specified, the key is extracted to the specified filename. Otherwise it is written to `stdout`.

```
$ saf-ssh-agent -x -f cozuser_saf.pub MY-RING:MY-CERT
```

Note: An administrator may export the key of a another user by prefixing the key ring name with `USERID/`. In order to do this, the administrator must have `UPDATE` access to the `IRR.DIGTCERT.LISTRING` SAF permission above.

Note: `READ` access to the `CLASS(FACILITY) IRR.DIGTCERT.LISTRING` resource allows the user to use any key ring the he or she owns. It is also possible to use ring-specific authorization, using `CLASS(RDATALIB)`. See the section *"Managing key rings and restricting access to them"* in *z/OS OpenSSH User's Guide* for more information.

3. Move a copy of the public key to the target system:

```
ZOS$ sftp -oPort=<port>  cozuser@linux1.myco.com
Connecting to n.n.n.n...
cozuser@linux1.myco.com's password:  ******
sftp>  ascii
Sets the file transfer type to ASCII.
sftp>  cd .ssh
sftp>  put -p cozuser_saf.pub authorized_keys
Uploading cozuser_saf.pub to /home/cozuser/.ssh/authorized_keys
cozuser_saf.pub                               100%  601     0.6KB/s   00:00
sftp> quit
```

Note: If you are adding more than one public key to `authorized_keys`, then you must log in to the remote system and append the new public key line to authorized_keys. Be careful that you don't replace an existing authorized_keys file.

Note: The `authorized_keys` file, the `.ssh` directory, and the home directory must not be writable by any user other than the owning userid. For details on required file permissions, see the section *"OpenSSH files Quick Reference / User-generated files"* in *z/OS OpenSSH User's Guide*

4. Using a SAF/RACF certificate for SSH authentication:
   - with Co:Z SFTP client:

```
ZOS$ cozsftp -k MY-RING:MY-CERT  cozuser@linux1.myco.com
```

(see also the SFTPSAMP or RUNSFTPK sample JCL)

- with Co:Z Launcher:

```
//COZCFG DD *
saf-cert=MY-RING:MY-CERT
```

(see also the RUNLNCHK sample JCL)

Renewing RACF self-signed certificates

You may wish to renew/extend a certificate used with OpenSSH, using the same self-signed key. The following commands can be executed by the owning user before the certificate expires. The owning user must have FACILITY authorities. Refer to "z/OS Security Server RACF Command Language Reference" for additional information.

```
DELETE 'SYSADM.CERT.REQ'

RACDCERT GENREQ(LABEL('MY-CERT'))                          +
            ID(COZUSER)                                    +
            DSN('SYSADM.CERT.REQ')

RACDCERT GENCERT('SYSADM.CERT.REQ')                        +
            ID(COZUSER)                                    +
            WITHLABEL('MY-CERT')                           +
            NOTAFTER(DATE(2016-12-31))                     +
            SIGNWITH(LABEL('MY-CERT'))
```

Appendix G. Client Compatibility

In general, SFTP clients that implement the SFTP specification [1] correctly work well with Co:Z SFTP Server. The following are general client functional areas that affect SFTP client compatibility with Co:Z SFTP server:

- Data set listings

  The SFTP "longname" field in the SSH_FXP_NAME packet is used by Co:Z SFTP to display z/OS specific formats for data sets, PDS members, and JES spool files. According to the SFTP specification, this field is not intended to be parsed by clients: *"clients SHOULD NOT attempt to parse the longname field for file attributes; they SHOULD use the attrs field instead."* Unfortunately, many clients do not follow these guidelines. The `unixls` setting can be used to cause Unix-style longname fields to be returned for z/OS resources. Refer to *Miscellaneous options* for more information.

- Data set name prefix (//)

  Co:Z SFTP server accepts two prefix strings to identify z/OS datasets as absolute paths. The first (//) is consistent with IBM's common usage. A secondary form (/-/) is also available. The secondary form is provided because some SFTP clients do not allow double slash characters to be sent. When using clients that do not support double slash characters, SFTP_ZOS_INITIAL_DIR can be set to /-/. Refer to *Section 2.2, "Configuring the Co:Z SFTP Server"* for more information on this optional setting.

- Using transfer options

  Some clients do not support interactive transfer options because new SSH/SFTP sessions are created (causing the interactive transfer options to be lost) when editing files on the server and/or when sending/receiving files. These clients are usually GUI clients with a transfer queue. Often these clients display a "connecting to the server" message when doing a file transfer, indicating that a new session has been created.

  The recommended configuration for these clients is to define transfer options in session config files on the Co:Z SFTP server (*Appendix C, Session config files*). Set the default transfer mode as text. This will ensure that the client can edit JCL members and other text files. For other cases, use file patterns. For example, add a pattern for *.pax to ensure that pax files are transferred with mode=binary. Refer to *Specifying file pattern specific options* for additional information.

- Unknown and estimated file sizes

  According to the SFTP specification, SSH_FILEXFER_ATTR_SIZE indicates whether the file attribute size field is present. Co:Z SFTP server ensures that this flag is off in cases where the data set size is unknown. There are also cases where the file size must be estimated. Clients are expected to send and receive files when the file size is unknown as well as when the size is an estimate. Clients that fail to support these cases may hang during the transfer or report a "file size mismatch" error. Clients that support SSH_FILEXFER_ATTR_SIZE correctly but do not support an estimated file size will work by setting the SFTP option **NOestsize**. Clients that do not support SSH_FILEXFER_ATTR_SIZE will not work with Co:Z SFTP server.

[1] SFTP specification: *http://tools.ietf.org/html/draft-ietf-secsh-filexfer-02*

The following summarizes how the file size is determined based on the file type:

- Posix file transfered with mode=binary: The actual file size is provided.
- Posix file transfered with mode=text: The actual file size is provided; however, some SFTP transfer options will cause the actual transfer size to differ. The transferred size will differ from the size provided when
  - the client and server codepages are not both multi-byte or not both single byte.
  - the `linerule` option changes the number of line separator characters during the transfer
  - an option such as `trim` or `pad` is used to modify the file during the transfer
- DASD data sets: An estimated file size is provided based on the used space, block size and logical record length.
- Tape data sets, GDG members, and PDS members: The provided size is unknown so the SSH_FILEXFER_ATTR_SIZE bit is not set indicating that the size is not specified.
- JES Spool Files: An estimated file size is provided based on the number of records.

Appendix H. Co:Z Environment Variables

The following table describes the environment variables defined by the Co:Z Toolkit. These variables can be set to override default behavior.

Table H.1. Environment variables

Variable	Context	Description
COZ_SERVER_KEEPALIVE	Co:Z Launcher	Interval in seconds Co:Z Server sends a NOOP packet. This option sends out actual data packets at the application level for situations where TCP_KEEPALIVE does not work due to firewall configuration. By default, this feature is not enabled.
COZ_SERVER_TCP_KEEPALIVE	Co:Z Launcher	Interval in seconds Co:Z Server sets the TCP_KEEPALIVE socket option. Note that this setting must be lower than the time that any firewall(s) may time out the connection. By default, this feature is not enabled.
COZ_SSH_CMD	Remote Dataset Pipes (Co:Z Target System Toolkit)	Specifies an alternate executable for the SSH client used to connect to z/OS. By default, this is `ssh`. For example, to use the PuTTY command line client `plink` instead of `ssh` set `COZ_SSH_CMD=/path/to/plink`.
COZ_SSH_OPTS	Remote Dataset Pipes (Co:Z Target System Toolkit)	Convenience setting for supplying SSH options, including userid and host when making remote dataset pipes calls. For example, the command `fromdsn -ssh user@host //mydsn` can be simplified to `fromdsn //mydsn` if `COZ_SSH_OPTS` is set to `user@host`. This is very handy for repeated use of the remote dataset pipes commands.
COZ_SSH_SUBSYS	Remote Dataset Pipes (Co:Z Target System Toolkit)	Specifies an alternate SSH server subsystem name for Dataset Pipes. By default, this is `dspipes`.
COZ_CLIENT_CODEPAGE	Remote Dataset Pipes (Co:Z Target System Toolkit)	Changes the default client code page, which is used for codepage translation in text mode data transfers (i.e. if the `-t` is not supplied). By default, the default client code page is set the result of the POSIX system call `nl_langinfo(CODESET)`.
COZ_DEFAULT_LOGSTREAM	Co:Z Log (all	Changes the default stream that the Co:Z Log facility

Variable	Context	Description
	contexts)	writes its messages to. By default, this is the `stderr` stream.
COZ_LOG	Co:Z Log (all contexts)	Sets default logging options for the Co:Z Log facility.
COZ_STRICT_CERT_CHECK	Co:Z Launcher, Co:Z SFTP	Affects the level of RACF digital certificate checking performed when authenticating. If set to true (the default), strict checking (e.g. certificate expiration date) is performed.
SFTP_LOGFILE	Co:Z SFTP	Pathname of file to where Co:Z SFTP log/debug messages are written. The default is `/tmp/sftp-server.<userid>.<...>.log`
SFTP_LOGDIR	Co:Z SFTP	Directory name (without trailing slash) where Co:Z SFTP log files are created, rather than /tmp or $TMPDIR. This variable is ignored if `SFTP_LOGFILE` is set.
SFTP_ZOS_OPTIONS	Co:Z SFTP	Used to set a default Co:Z SFTP options string for the user. There is no default. Example: `SFTP_ZOS_OPTIONS=mode=text,l=crlf`. To set Co:Z SFTP Server options, this variable is exported in the user's `sftp-server.rc` file. To set Co:Z SFTP client options, export this environment variable prior to running **cozsftp**

Appendix I. Restricting OpenSSH users to SFTP

The common technique for restricting ssh capabilities is to change the user's default shell (the "default program" in the OMVS segment) to a shell that only allows certain commands and no interactive access. The sample script below can be customized and used as the user's "restricted shell". Put this script somewhere in your Co:Z bin install directory and make its permissions `u=rwx,go=rx` (i.e 755). Use the full path name of the script as the users shell. Make sure that the script uses full path names. You can remove the `dspipes.sh` entry from the list if you don't want to allow remote Dataset Pipes commands via ssh.

```
#!/bin/sh
# A shell script which can be set as a users default shell
# to only allow certain commands or ssh subsystems to run,
# disallowing full shell logins.

if test $# -ge 2 -a "$1" = "-c"
then
 case $2 in
 # Update this list to match what you have in /etc/ssh/sshd_config
 # or add any other commands that you would like to allow
 # from ssh.
 /opt/dovetail/coz/bin/sftp-server.sh) exec $2;;
 /opt/dovetail/coz/bin/dspipes.sh) exec $2;;
 *);;
 esac
fi

# Write out whatever messages you want your users to see
# if they try something else
echo "Only sftp and dataset pipes file transfers are allowed from this account."
exit 1
```

Note: this will not only restrict the user from using an interactive shell under OpenSSH, but will prevent them from running an interactive shell under TSO as well. Also, any batch jobs that run with their userid will also use this as the default Unix shell (BPXBATCH or COZBATCH).

Appendix J. Setting up a test OpenSSH system on z/OS

It's sometimes convenient to create your own z/OS SSHD server on an alternate port for testing purposes. You can do this without any special privileges, and the SSHD server will run fine, except that it will only allow logins for the userid that it is running under.

This is especially handy if your Systems Programmer doesn't understand immediately that adding an SSH user subsystem doesn't introduce any new security risks.

Procedure J.1. General outline for adding a test SSHD server

1. Create your own ssh directory, say ~/sshd, and copy the file `/etc/ssh/sshd_config` into it:.

```
zos$ mkdir ~/sshd
zos$ cp /etc/ssh/sshd_config ~/sshd
```

2. In this directory, generate your DSA and RSA host keys, as directed in the *z/OS OpenSSH User's Guide*.

   If you can copy the keys in /etc/ssh directory, then you will avoid "host key" mismatch problems if you switch your SSH client from the production to the test server. If you do copy the production host keys, make sure that you change the file permissions to `600` so that they can't be read by others.

3. Edit your copy of sshd_config:

   a. Find the line "Subsystem" which defines the sftp subsystem

   b. Add a new line after this line:

```
Subsystem dspipes <COZ_INST>/bin/dspipes
```

   (where <COZ_INST> is the directory where Co:Z Toolkit is installed).

   c. Uncomment the `Port` line and set it to an available port

   d. Uncomment / add the following lines (to use the private keys generated in the previous step):

```
HostKey ./ssh_host_rsa_key
HostKey ./ssh_host_dsa_key
```

   (where <COZ_INST> is the directory where Co:Z Toolkit is installed).

4. From a z/OS shell, change to the directory that you created and start your copy of SSHD:

```
/usr/sbin/sshd -e -D -f ./sshd_config
```

Note: If you are unable to execute `/usr/sbin/sshd`, you may be able to copy it to your local directory, add the execute bit (`chmod +x ~/sshd/sshd`) and run the above command using this local copy.

5. To connect to your test SSHD server from a client, don't forget to use the `-ssh -p port` SSH option on your ssh, **fromdsn** or **todsn** commands.

Appendix K. Creating a Custom Unicode Table from the IBM FTP Translate Table

Co:Z SFTP provides transfer options for specifying client (`clientcp`) and server (`servercp`) code pages to be used during file transfers. Additionally, the `technique` option can be used to override the default Unicode Services value of LMREC. The following are condensed instructions for adding the IBM FTP translate table to Unicode Services. The IBM FTP translate table can be used as is or modified to meet your specific custom unicode table requirements.

Note

The condensed instructions below are for illustrating how the Co:Z SFTP `clientcp`, `servercp` and `technique` transfer options are used. Refer to IBM's Unicode Services User's Guide and Reference (SA22-7649) for complete information on the IBM supported method for adding new conversion tables to Unicode Services.

Note

Alternatively, if using CoZ: SFTP 2.3.0 or higher, the `trtab` transfer option can be used to specify the translate table to use for text mode transfers. The IBM FTP translate table can be specified using `trtab=STANDARD` in place of the `clientcp`, `servercp`, and `technique` options. Refer to *General transfer options* for additional information on the `trtab` option.

Adding the IBM FTP translate table to Unicode Services:

- Rather than modifying `SYS1.SCUNTBL`, create a new PDS with your own high level qualifier: `HLQ.SCUNTBL` with `RECFM=F,LRECL=256,BLKSIZE=256`
- Create new members in `HLQ.SCUNTBL`

  Codepages IBM-850 and IBM-037 seem to be the closest to the IBM FTP translate table. Using IBM's Unicode Services User's Guide and Reference (SA22-7649), the two-character codes for IBM-850 and IBM-037 are **EB** (850) and **AA** (037), respectively. Select a new conversion technique code, in this example, **"2"** is used. This information will be used to name the new members added to `HLQ.SCUNTBL`.

  View `TCPIP.STANDARD.TCPXLBIN` in hex and verify that the translation meets or is close to meeting your requirements. `TCPIP.STANDARD.TCPXLBIN` has the following three records:

  - a comment
  - the 256-byte ASCII->EBCDIC table
  - the 256-byte EBCDIC->ASCII table

  Create a new member `CUN2EBAA` in the `HLQ.SCUNTBL` dataset (850->037, TECH=2). Copy the first

non-comment record from `TCPIP.STANDARD.TCPXLBIN` into this member.

Create another new member `CUN2AAEB` (037->850, TECH=2). Copy the second non-comment record from `TCPIP.STANDARD.TCPXLBIN` into this member.

- Add a `CUNUNIxx` member to PARMLIB (choose your preferred suffix).

```
REPLACE FROM(037) TO(850) TECH(2) DSN(HLQ.SCUNTBL)
REPLACE FROM(850) TO(037) TECH(2) DSN(HLQ.SCUNTBL)
```

- Activate the new `CUNUNIxx` PARMLIB member using the SET command. The following shows the command and resulting messages.

```
SET UNI=XX
IEE252I MEMBER CUNUNIXX FOUND IN XXXX.PARMLIB
IEF196I IGD103I SMS ALLOCATED TO DDNAME SYS00013
IEF196I IGD104I HLQ.SCUNTBL            RETAINED,
IEF196I DDNAME=SYS00013
IEF196I IGD103I SMS ALLOCATED TO DDNAME SYS00014
```

- Once you have tested the new translate table, add UNI=xx to your `IEASYSxx` parmlib member so that table is available permanently.

Testing the new table added to Unicode Services:

- The ***showtrtab*** command can be used to display the new translate table. Refer for to the *Co:Z Dataset Pipes Command Reference* for additional information.

```
>showtrtab -s IBM-850 -t IBM-037 -q 2
00:  00 01 02 03   37 2D 2E 2F   16 05 25 0B   0C 0D 0E 0F
10:  10 11 12 13   3C 3D 32 26   18 19 3F 27   22 1D 35 1F
20:  40 5A 7F 7B   5B 6C 50 7D   4D 5D 5C 4E   6B 60 4B 61
30:  F0 F1 F2 F3   F4 F5 F6 F7   F8 F9 7A 5E   4C 7E 6E 6F
40:  7C C1 C2 C3   C4 C5 C6 C7   C8 C9 D1 D2   D3 D4 D5 D6
50:  D7 D8 D9 E2   E3 E4 E5 E6   E7 E8 E9 AD   E0 BD 5F 6D
60:  79 81 82 83   84 85 86 87   88 89 91 92   93 94 95 96
70:  97 98 99 A2   A3 A4 A5 A6   A7 A8 A9 C0   4F D0 A1 07
80:  00 01 02 03   37 2D 2E 2F   16 05 25 0B   0C 0D 0E 0F
90:  10 11 12 13   3C 3D 32 26   18 19 3F 27   22 1D 35 1F
A0:  40 5A 7F 7B   5B 6C 50 7D   4D 5D 5C 4E   6B 60 4B 61
B0:  F0 F1 F2 F3   F4 F5 F6 F7   F8 F9 7A 5E   4C 7E 6E 6F
C0:  7C C1 C2 C3   C4 C5 C6 C7   C8 C9 D1 D2   D3 D4 D5 D6
D0:  D7 D8 D9 E2   E3 E4 E5 E6   E7 E8 E9 AD   E0 BD 5F 6D
E0:  79 81 82 83   84 85 86 87   88 89 91 92   93 94 95 96
F0:  97 98 99 A2   A3 A4 A5 A6   A7 A8 A9 C0   4F D0 A1 07
```

- The following commands can also be used to test the new translate table.

```
lzopts mode=text,c=IBM-037,s=IBM-850,technique=2,l=none
put //HLQ.TEST.DATA custom_table_test.txt
```

Appendix L. License

The Co:Z Co-Processing Toolkit for z/OS, comprised of Co:Z Launcher, Co:Z Dataset Pipes, Co:Z SFTP, Co:Z Batch, Co:Z ssh-proxyc and Co:Z Target System Toolkit (in object code form only) is distributed under the Co:Z Community License Agreement (see below). *Note:* This community license is superseded for Co:Z Toolkit Enterprise License and Support customers. All components are distributed in binary form.

```
COMMUNITY LICENSE AGREEMENT

PLEASE READ THIS COMMUNITY LICENSE AGREEMENT (THIS “AGREEMENT”) CAREFULLY.
THIS AGREEMENT SETS FORTH THE TERMS ON WHICH DOVETAILED TECHNOLOGIES, LLC
(“DOVETAIL”), A MISSOURI LIMITED LIABILITY COMPANY, MAKES AVAILABLE THE
CO:Z CO-PROCESSING TOOLKIT FOR z/OS AT NO CHARGE FOR DOWNLOAD, INSTALLATION AND
USE BY THE COMMUNITY. BY DOWNLOADING, INSTALLING, OR USING THE SOFTWARE, YOU
ACKNOWLEDGE THAT YOU HAVE READ, UNDERSTAND, AND AGREE TO BE LEGALLY BOUND BY
THIS AGREEMENT.

1. DEFINITIONS. As used in this Agreement, the following capitalized terms
shall have the following meanings:

“Documentation” means Dovetail’s accompanying user documentation for the
Software, as may be updated by Dovetail from time to time, in print or
electronic form.

“Software” means the Co:Z Co-Processing Toolkit for z/OS, comprised of Co:Z
Launcher, Co:Z Dataset Pipes, Co:Z SFTP, Co:Z Batch, Co:Z ssh-proxyc and Co:Z
Target System Toolkit, in object code form only, together with certain sample
code and scripts in source form.

“Update” means any bug fix, enhancement, or other modification to or update for
the Software issued by Dovetail for general release to the Software community.

“You” means the person or entity downloading, installing or using the Software.
If you are downloading, installing or using the Software on behalf of a company
or organization, the term “You” refers to both you and your company or
organization, and you represent and warrant that you have authority to bind
your company or organization to the provisions hereof.

2. SOFTWARE LICENSE. During the term of this Agreement, and subject to the
provisions hereof, Dovetail hereby grants to You, and You hereby accept, an
enterprise-wide, non-exclusive, non-transferable, royalty-free and fully
paid-up license to install and use the Software on an unlimited number of Your
servers, solely for Your internal business purposes, in accordance with the
Documentation, and in compliance with all applicable laws and regulations.

3. LICENSE RESTRICTIONS. You may not install or use the Software for any
purpose other than as expressly authorized under Section 2. Without limiting
the foregoing, You shall not, nor shall You authorize any other person or
entity to: (a) distribute, rent, lease, lend, sell, sublicense or otherwise
make the Software available to any third party; (b) modify, adapt, alter,
translate, or create derivative works of the Software; (c) use the Software
in or as part of a service bureau, timesharing or outsourcing capacity,
including to extend the Software to or manage, operate, or support the Software
for third parties; (d) develop an alternative to the Software that is based on
or derived from, in whole or in part, the Software or Documentation; (e) use
the Software in violation of any applicable laws or regulations; (f) remove or
```

```
obscure any copyright, trademark or other proprietary rights notices or
designations on the Software, the Documentation or any copies thereof; or (g)
reverse engineer, decompile, disassemble, or otherwise attempt to derive the
source code for the Software, except where such reverse engineering is
expressly permitted under applicable law, but then only to the extent that
Dovetail is not entitled to limit such rights by contract.

4. UPDATES. From time to time, Dovetail may make available Updates for the
Software as a general release to the Software community. All such Updates
(whether posted by Dovetail on the Dovetail website or included with the
Software) shall be deemed part of the Software, and are licensed to You under
the license and other provisions of this Agreement, together with any
supplementary license terms that Dovetail may provide for such Updates.
Notwithstanding the foregoing, Dovetail reserves the right to amend, supplement
or replace the terms of this Agreement in connection with Updates to or new
versions of the Software, and in such case, the terms accompanying such Update
or new version will control.

5. YOUR RESPONSIBILITIES. You are responsible for: (i) installation of the
Software and any Updates; (ii) selecting and maintaining all third party
hardware, software, peripherals and connectivity necessary to meet the system
requirements for the Software; (iii) creating a restore point for Your systems
and backing up and verifying all data; and (iv) adopting reasonable measures to
ensure the safety, security, accuracy and integrity of Your facilities,
systems, networks and data. Dovetail shall have no responsibility or liability
arising out of or resulting in whole or in part from Your failure or delay
performing any such responsibilities, or for acts or omissions of third
parties, Internet or telecommunications failures, or force majeure or other
events beyond Dovetail’s reasonable control.

6. SUPPORT. This Agreement does not include, and Dovetail shall have no
obligation under this Agreement to provide, any technical support or other
professional services for the Software. If You are interested in purchasing a
support plan for the Software, You should visit the Dovetail website to review
Dovetail’s then current offerings.

7. TERM; TERMINATION. This Agreement and Your license rights hereunder shall
continue unless and until terminated as set forth herein. You may terminate
this Agreement for convenience at any time by uninstalling, erasing all copies
of, and ceasing all use of the Software and Documentation. This Agreement shall
terminate immediately and automatically if You violate the license terms or
restrictions for the Software, or materially breach any other provision of this
Agreement and fail to cure such breach within ten (10) days after receiving
notice thereof from Dovetail. Upon the expiration or termination of this
Agreement for any reason: (i) Your license to the Software shall automatically
and immediately terminate; and (ii) You shall discontinue use of the Software,
promptly (within 5 days) uninstall and remove any remnants of the Software and
Documentation from Your computers, network and systems, and destroy (or return
to Dovetail) all tangible copies of the Software and Documentation in Your
possession. Sections 1, 3, 5, 7, 8, 9, 10 and 11 of this Agreement shall
survive the expiration or termination of this Agreement for any reason, and
shall be binding on and inure to the benefit of the parties and their permitted
successors and assigns.

8. DISCLAIMER. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED TO YOU UNDER THIS
AGREEMENT “AS IS” WITHOUT REPRESENTATIONS OR WARRANTIES OF ANY KIND, EXPRESS
OR IMPLIED, AND ALL USE IS AT YOUR OWN RISK. WITHOUT LIMITING THE FOREGOING,
DOVETAIL AND ITS SUPPLIERS HEREBY DISCLAIM ANY IMPLIED OR STATUTORY WARRANTIES
OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, OR
```

```
NON-INFRINGEMENT. THE SOFTWARE IS NOT INTENDED OR LICENSED FOR USE IN ANY
HAZARDOUS OR HIGH-RISK ACTIVITY. DOVETAIL DOES NOT WARRANT THAT THE SOFTWARE
WILL OPERATE UNINTERRUPTED OR ERROR-FREE, OR MEET YOUR BUSINESS, TECHNICAL
OR OTHER REQUIREMENTS. NO EMPLOYEE OR AGENT HAS AUTHORITY TO BIND DOVETAIL TO
ANY REPRESENTATIONS OR WARRANTIES NOT EXPRESSLY SET FORTH IN THIS AGREEMENT.

9. PROPRIETARY RIGHTS. Dovetail and its suppliers shall retain exclusive right,
title and interest in and to the Software, including the object code, source
code, program architecture, design, coding methodology, Documentation, screen
shots, and “look and feel” therefor, all Updates thereto, all goodwill
associated therewith, and all present and future copyrights, trademarks, trade
secrets, patent rights and other intellectual property rights of any nature
throughout the world embodied therein and appurtenant thereto. All rights and
licenses to the Software not expressly granted to You in this Agreement are
reserved by Dovetail and its suppliers. From time to time, You may submit
suggestions, requests or other feedback for the Software. Dovetail shall be
free to commercialize and use such feedback, including for developing
improvements to its products and services, free of any claims, payment
obligations, or proprietary, confidentiality or other restrictions of any kind.

10. LIMITATIONS ON LIABILITY. IN NO EVENT SHALL DOVETAIL BE LIABLE FOR ANY
INDIRECT, INCIDENTAL, CONSEQUENTIAL, EXEMPLARY, SPECIAL, PUNITIVE, OR SIMILAR
DAMAGES ARISING OUT OF OR RELATED TO THE SOFTWARE OR THIS AGREEMENT, INCLUDING
LOSS OF BUSINESS, PROFITS OR REVENUE, LOSS OR DESTRUCTION OF DATA, BUSINESS
INTERRUPTION OR DOWNTIME. THE TOTAL CUMULATIVE LIABILITY OF DOVETAIL ARISING
OUT OF AND RELATED TO THE SOFTWARE AND THIS AGREEMENT SHALL NOT, REGARDLESS OF
THE NUMBER OF INCIDENTS OR CAUSES GIVING RISE TO ANY SUCH LIABILITY, EXCEED TEN
U.S. DOLLARS ($10). THE LIMITATIONS ON LIABILITY IN THIS SECTION SHALL APPLY TO
THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, REGARDLESS OF THE CAUSE OF
ACTION OR BASIS OF LIABILITY (WHETHER IN CONTRACT, TORT OR OTHERWISE), EVEN IF
ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THESE LIMITATIONS ON LIABILITY ARE
AN ESSENTIAL PART OF THIS AGREEMENT, AND SHALL BE VALID AND BINDING EVEN
IF ANY REMEDY IS DEEMED TO FAIL OF ITS ESSENTIAL PURPOSE.

11. MISCELLANEOUS

Governing Law. This Agreement shall be governed and interpreted for all
purposes by the laws of the State of Missouri, U.S.A., without reference to any
conflict of laws principles that would require the application of the laws of a
different jurisdiction. The United Nations Convention on Contracts for the
International Sale of Goods and the Uniform Computer Information Transactions
Act (as enacted in any jurisdiction) do not and shall not apply to this
Agreement, and are hereby specifically excluded.

Jurisdiction; Venue. Any dispute, action or proceeding arising out of or
related to this Agreement shall be commenced in the state courts of St. Louis
County, Missouri or, where proper subject matter jurisdiction exists, the
United States District Court for the Eastern District of Missouri. Each party
irrevocably submits to the personal jurisdiction and exclusive venue of such
courts, and waives any objections thereto, including based on forum non
conveniens.

Notices. All notices under this Agreement shall be in writing, and shall be
delivered personally or by postage prepaid certified mail or express courier
service, return receipt requested. Notices to You may be delivered to the most
current address on file. Notices to Dovetail shall be directed to the following
address, unless Dovetail has provided an alternative notice address:

         Dovetailed Technologies, LLC
```

```
         305 Willowpointe Drive
         St. Charles, MO 63304

Assignments. You may not assign or transfer this Agreement, or any rights or
duties hereunder, in whole or in part, whether by operation of law or
otherwise, without the prior written consent of Dovetail. Any attempted
assignment or transfer in violation of the foregoing shall be null and void
from the beginning and without effect. Dovetail may freely assign or transfer
this Agreement, including to a successor upon Dovetail’s merger, acquisition,
corporate reorganization, or sale or other transfer of all or substantially all
of its business or assets to which this Agreement relates.

Relationship; Third Party Beneficiaries. The parties hereto are independent
contractors. Nothing in this Agreement shall be deemed to create any agency,
employment, partnership, fiduciary or joint venture relationship between the
parties, or to give any third party any rights or remedies under or by reason
of this Agreement; provided, however, the disclaimers and limitations on
liability in this Agreement shall extend to Dovetail and its directors,
officers, shareholders, employees, agents, and affiliates. All references to
Dovetail in connection therewith shall be deemed to include the foregoing
persons and entities, who shall be third party beneficiaries of such
contractual disclaimers and limitations and entitled to accept all benefits
afforded thereby.

Equitable Relief. The Software comprises the confidential and proprietary
information of Dovetail and its suppliers, and constitutes a valuable trade
secret. You acknowledge that Your breach of the license or ownership provisions
of this Agreement would cause irreparable harm to Dovetail, the extent of which
would be difficult and impracticable to assess, and that money damages would not
be an adequate remedy for such breach. Accordingly, in addition to all other
remedies available at law or in equity, and as an express exception to the
jurisdiction and venue requirements of this Agreement, Dovetail shall be
entitled to seek injunctive or other equitable relief in any court of competent
jurisdiction.

U.S. Government Restricted Rights. The Software and Documentation are licensed
with RESTRICTED RIGHTS as "Commercial Items," as that term is defined at 48
C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial
Computer Software Documentation," as such terms are used in 48 C.F.R. §12.212
or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48
C.F.R. §227.7202-1 through 227.7202-4, as applicable, the Commercial Computer
Software and Commercial Computer Software Documentation is licensed (if at all)
to U.S. Government end users only as Commercial Items, and with only those
rights as are granted to other licensees pursuant to this Agreement.

Export Control. The Software and underlying information and technology may not
be accessed or used except as authorized by United States and other applicable
law, and further subject to compliance with this Agreement. The Software may
not be exported or re-exported into any U.S. embargoed countries, or to anyone
on the U.S. Treasury Department's list of Specially Designated Nationals or the
U.S. Department of Commerce Denied Person’s List or Entity List. You represent
and warrant that You and Your end users are not located in, under the control
of, or a national or resident of any country or on any such list.

Amendment; Waiver. This Agreement may be amended only by a written instrument
signed by an authorized representative of Dovetail. No rights shall be waived
by any act, omission, or knowledge of a party, except by an instrument in
writing expressly waiving such rights and signed by an authorized
representative of the waiving party. Any waiver on one occasion shall not
```

```
constitute a waiver on subsequent occasions.

Severability; Construction. If any provision of this Agreement is determined to
be invalid or unenforceable under applicable law, such provision shall be
amended by a court of competent jurisdiction to accomplish the objectives of
such provision to the greatest extent possible, or severed from this Agreement
if such amendment is not possible, and the remaining provisions of this
Agreement shall continue in full force and effect. The captions and section
headings in this Agreement are for reference purposes only and shall not affect
the meaning or interpretation of this Agreement. The term “including” as used
herein means “including without limitation.” The terms “herein,” “hereto,”
“hereof,” and similar variations refer to this Agreement as a whole, rather
than to any particular section.

Entire Agreement. This Agreement sets forth the entire agreement of the parties
and supersedes all prior agreements and understandings, whether written or
oral, with regard to the subject matter hereof. Any additional or conflicting
terms proposed by You in any purchase order, request for proposal,
acknowledgement, or other writing shall not be binding, and are hereby objected
to and expressly rejected.
```