Co:Z® Co-Processing Toolkit for z/OS

Co:Z Launcher - User's Guide

V 4.5.0 Edition

Published July, 2017

Table of Contents

1. Introduction

The Co:Z Launcher is a batch utility which remotely launches a process on a distributed system, redirecting input and output from that process to traditional z/OS datasets or spool files. Remote processes are securely launched using proven SSH (Secure Shell) technology to the target platform, which may be Linux, Windows, or other Unix/POSIX environments.

1.1 Co:Z Launcher Features

- Securely launch and control remote processes (programs, scripts, etc.) from a z/OS batch job step or started task.
- Redirect input and output of remote process to DDs in the launching job step.
- Target process exit code is captured as job step condition code.
- Co:Z Launcher job step acts as a server for z/OS dataset I/O.
- z/OS console commands can be used to monitor, control, and send input to remote process.
- Existing z/OS scheduling and automation facilities can be used to schedule, monitor, and control processes on all servers on the network.
- Dataset Pipes client commands may be used in the target process to reach back and access datasets in the launching jobstep. These commands provide flexible conversion of z/OS datasets to streams for use in target applications. Options allow for control of line rules, translation, padding/truncation, dataset allocation and DCB processing.
- SAF/RACF Digital Certificates may be used for client authentication.

2. Co:Z Launcher Installation

In order to use the Co:Z Launcher to enable z/OS batch jobs to remotely launch a process on a distributed system, installation is required for the *Co:Z Toolkit for z/OS*. Additionally, the *Co:Z Target System Toolkit* must be installed on the remote systems that you have identified.

2.1 Co:Z Launcher environment requirements

- **z/OS requirements**
  - Co:Z Toolkit for z/OS
  - Batch job userid allowed to listen on local port; OMVS segment required
- **Target System requirements:**
  - Co:Z Target System Toolkit
  - OpenSSH sshd
  - sshd_config `AllowTcpForwarding=yes` for target userid

2.2 Co:Z Launcher Quick Start

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

On z/OS:

1. Edit `//COZUSER.COZ.SAMPJCL(COZCFGD)`, updating the `server-path` variable with your installation directory.
2. Edit `//COZUSER.COZ.SAMPJCL(COZPROC)`, updating the `LIBRARY` and `COZCFGD` arguments with your installation datasets.
3. Verify that `/etc/ssh/sshd_config` has `AllowTcpForwarding=yes`. If not set, update the sshd configuration. Restart SSHD with the following:

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

4. Test that the z/OS user running Co:Z Launcher jobs can SSH to the target system without an interactive password.

```
ssh user@linux1.myco.com
```

   See *OpenSSH keypair authentication* for additional information, if needed.

5. Create and test JCL similar to the example provided here: *Running with an OpenSSH keypair*. This JCL is also included in `//COZUSER.COZ.SAMPJCL(RUNLNCH)`.

6. Once this test job runs successfully, experiment by trying *Co:Z Launcher Examples*. See *General Dataset Pipes Examples* for some Dataset Pipes commands to add to your JCL.

3. Co:Z Launcher Configuration

The Co:Z Launcher is initiated in batch via JCL job steps that execute the `COZLNCH` load module. The z/OS installation package includes a sample stored procedure for invoking the launcher `COZPROC`. The launcher is configured through a set of customizable properties, which are described below (default values are shown in braces).

3.1 Co:Z Launcher Properties

Some server properties (`server-ports`, `server-ip-stack` and `server-host`) may be optionally suffixed with a z/OS sysid. In this case, these properties will apply only to a specific z/OS system. This allows for a single COZCFG member to be used for all of the candidate z/OS systems in an installation.

The PDS member `COZCFGD` can be customized for each installation to provide system level defaults for many of these properties.

Note

If duplicate property names are supplied, only the final value is used. To specify multi-line values for a property, place a backslash (\) continuation character on the line(s) to be continued.

Required Properties

Each installation is required to customize the following properties:

server-path {`<COZ_INST>/cozserver`}
: The absolute path on the server of the CozServer executable, where <COZ_INST> is the directory where Co:Z Toolkit is installed.

server-ports[-sysid] {`none`}
: No longer a required option as of release 4.0.0.

Optional Properties

The following properties may be overidden in COZCFGD or by individual job step

server-ports[-sysid] {`none`}
: The range of ports reserved for communication between CoZServer and the target system. This option should only be used for a Co:Z Launcher release older than 4.0.0 or when option ssh-forward-dynamic-port is required to be `false`. When specified, each invocation of a Co:Z Launcher batch job will find one available port in this range and establish a socket listener.

  If ssh-tunnel=true (the default), an available port in this range will be bound to the z/OS loopback adapter (127.0.0.1), and the target program on the target server will connect to this port via the tunnel established by ssh.

  If ssh-tunnel=false, an available port in this range will be bound to any stack on z/OS (this can be changed using the server-ip-stack property), and the target program on the target server will connect to this port directly over the network.

Installations must reserve a port range on z/OS large enough for each concurrent Co:Z Launcher batch job. If ssh-tunnel=true, then the target servers must also ensure that these port are available (unless ssh-forward-dynamic-port=true). If multiple z/OS systems share the same target machines, each z/OS system should reserve its own port range.

The following example sets up a 20 port pool for use by any Co:Z Launcher instance.

```
server-ports=8040-8059
```

The following example sets up separate 20 port pools for three z/OS systems running in an installation (and sharing the same COZCFGD member). If ssh-tunnel=true (the default), then each target system must make 8040-8099 available.

```
server-ports-SYSA=8040-8059
server-ports-SYSB=8060-8079
server-ports-SYSC=8080-8099
```

ssh-le-options {`none`}
: Custom Language Environment (LE) options to set for the ssh client process created by the Launcher. No options are set by default, but see the COZCFGD sample for the recommended options to work around a problem that causes out-of-memory conditions in Ported Tools OpenSSH. See IBM APAR OA34819.

ssh-options {`none`}
: Additional options to be supplied to z/OS ssh command.

ssh-path {`/bin/ssh`}
: Specifies the location of the z/OS ssh client executable.

ssh-tunnel {`true`}
: If true, target program IO requests (via fromdsn and todsn) are tunnelled over ssh via reverse port forwarding. If false, direct socket connects are made to the server.

ssh-forward-dynamic-port {`true`}
: If true, allows the target ssh server to dynamically assign the target port. In order to enable this option:

  - The ssh-tunnel option must be set to true
  - IBM Ported Tools OpenSSH v1.3 / z/OS V2R2 OpenSSH or higher is required
  - The target ssh server must support dynamic reverse port forwarding (e.g. OpenSSH version 5.3 or later)

  This option was introduced in release 3.6.0 with a default value of `false`. The default was changed to `true` in release 4.0.0 (but note that the default remains `false` if the server-ports property is set). The intent of this change is to enable an ephemeral port to be used on z/OS by default, eliminating the need for reserved port ranges. The server-port option is no longer a required property as of release 4.0.0.

  If false, the target port is set matching the selected server-port. Setting the target port to the server-port value

can cause collisions when multiple Co:Z Launcher jobs from different LPARS are running concurrently to the same target host. Enabling the ssh-forward-dynamic-port option prevents these collisions.

ssh-forward-dynamic-port-wait {`30`}
The number of seconds the Launcher will wait for the target system to report its dynamic port.

saf-cert {`none`}
Specifies that the user's RACF Digital Certificate should be used for client authentication. The value supplied for this property is in the form KEYRING[:LABEL]. If LABEL is omitted, the keyring's default label will be used. Examples:

```
saf-cert=MY-RING
saf-cert=MY-RING:MY-CERT
```

agent-path {`/opt/dovetail/coz/bin/cozagent`}
The executable path on the target of the CozAgent executable. Note that the client make install target places the Co:Z executables at `/opt/dovetail/coz/bin` by default.

agent-options {`none`}
Command line options to CoZAgent. These include:

- `-c` -- allow the operator to communicate with the agent to control the target program. See *Section 3.3, "Console communication"* for a list of available commands.
- `-n` -- run the target-command without a shell.

agent-output-wto {`false`}
If true, messages written by the CoZAgent are written to the operator console. If false, they are written to the launcher's stdout (DD://SYSPRINT)

server-host[-sysid] {`gethostname()`}
The external address of the CoZServer running on z/OS. If ssh-tunnel=false, the target program will connect to this address. If ssh-tunnel=true, this value is ignored.

server-ip-stack[-sysid] {`0.0.0.0 (all addresses)`}
The IP address the CoZServer will accept connections on. If ssh-tunnel=true, this value is ignored.

server-env-MY_VAR {`none`}
Customized server environment variables that will be set prior to launching the CoZServer. MY_VAR should be replaced by the name of the environment variable to be set. These environment variables will also be adopted by the Launcher itself. See *Appendix Co:Z Environment Variables* for a list of environment variables.

target-env-MY_VAR {`none`}
Customized target environment variables that will be set prior to launching the target program. MY_VAR should be replaced by the name of the environment variable to be set.

target-command {`none`}
The target program to be run by CoZAgent. If not supplied, the target user's default shell will be executed.

target-host {`none`}
: The hostname or IP address of the target machine. This value and target-user may alternatively be supplied in the form `user@host:port` on the COZPROC ARGS= parameter.

target-user {`none`}
: The userid that the target program runs under on the target machine. This value and target-host may alternatively be supplied in the form `user@host:port` on the COZPROC ARGS= parameter.

properties-exit {`none`}
: Specifies the executable Unix command and arguments that are used to run a Unix program or shell script that may write additional configuraton properties to its **stdout**. Output lines from this program will be used as additional configuration properties as if they were specified at the end of the **DD:COZCFG** file. A practical use for this feature might be to dynamically determine the **target-host** property from a list of candidate servers.

  The command string specified is run using **/bin/sh -c "command args"**. Note that the Co:Z Launcher batch utility does not run a "login" shell, so that the `PATH` environment variable will only contain `/bin` and other variables as determined by the installations `/etc/init.options` file. Therefore, a fully qualified command path name is often required, and a shell script may wish to "dot in" `/etc/profile` and `~/.profile` if appropriate.

3.2 Configuration Best Practices

Data transferred between z/OS and the target remote system can be configured in one of two ways:

- tunnelled (and encrypted) over ssh via reverse port forwarding
- direct socket connection (better throughput with lower overhead, but unencrypted)

The preferred configuration for multiple concurrent jobs is to use tunnelling with dynamic port forwarding.

When tunneling

When using tunneling, the preferred configuration is to use dynamic port forwarding when supported. For a tunneled configuration, use the following settings (default in COZCFGD for release 4.0.0 and later):

```
ssh-tunnel=true
ssh-forward-dynamic-port=true
#server-ports=
```

If not using IBM Ported Tools 1.3 (or later) or if dynamic port forwarding is not supported on the target system, override the defaults in a job specific COZCFG:

```
ssh-forward-dynamic-port=false
server-ports=8040-8059
```

When not tunneling

When using a direct socket connection and no `server-ports` definition, a server ephemeral port will be used

which may cause trouble with existing firewall policies. Therefore, it is recommended that a `server-ports` range be defined so that network administration has visibility to the ports being used by Co:Z. The following settings are recommended (note that the `ssh-forward-dynamic-port` option will be ignored, as it only takes effect if `ssh-tunnel=true`):

```
ssh-tunnel=false
server-ports=8040-8059
```

3.3 Console communication

If the CoZAgent is started with the console communication switch (-c), the MVS system console can be used to communicate with the target system. This interaction can occur once the target program has completed its processing of the STDIN DD, if it exists. In this case, the target program will not receive an EOF from stdin until a /QUIESCE command is sent from the console. The available commands are described below.

If the CoZAgent is started *without* the -c switch, no console communication is permitted. When the target program finishes reading STDIN, it will receive an EOF as in normal processing.

Console commands are sent to the remote agent by using the MVS `MODIFY (F)` and `STOP (P)` commands. The modify string must be prefixed by keyword `APPL=`.

If the text supplied on the modify command is surrounded by single quote (') characters, it is passed unmodified to the console. Note that in some cases ISPF panels will force entered text to uppercase. If so, eliminate the single quote characters and the entire command will be folded to lower case by Co:Z, which is generally more compatible with remote Unix systems. In this situation, upper case characters may be specified by prefixing each character with an underscore ('_').

The MVS console supresses certain characters, such as `` `,\,~,^,[,],{,} ``. These characters should not be specified.

Note

Because of the way z/OS Unix names child processes, your job/task name should consist of 7 characters or less (or use an identifier) if you wish to use console commands. If you use an 8 character jobname, you will see the following message `IEE342I MODIFY REJECTED-TASK BUSY` (the command will still be processed).

Commands directed to the CoZAgent process

/QUIESCE
: Sends an EOF to the target program's stdin. This will allow the target program that waits for interactive stdin commands to perform its normal completion processing.

/KILL [signal_level] {`SIGKILL`}
: Issues the specified signal to the target program.

/CMD <command>
: Issues `command` as a system() call. Typical commands include process status commands such as `ps -eaf`. Any resulting stdout data is written either to the MVS console or the STDOUT DD, depending on the value of

the `agent-output-wto` property, described above.

Any console command `not` prefixed with a slash (/) as above is sent directly to the target program for processing.

Examples:

```
F MYJOB,APPL=/CMD PS -E_AF
F MYJOB,APPL=INPUT TO REMOTE PROGRAM
F MYJOB,APPL=/QUIESCE
```

4. Running the Co:Z Launcher

This chapter explains how to run the Co:Z Launcher, based on the user's configured authentication mechanism. 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 A, Client Authentication Mechanisms*. Note that instructions in this appendix must be followed in order to run the examples described below.

4.1 Running with SSH_ASKPASS Authentication

Note: The JCL discussed below is included in the Co:Z toolkit samples as member RUNLNCHP

```
//USERP JOB (),'DOVETAIL',MSGCLASS=H,NOTIFY=&SYSUID
//PROCLIB JCLLIB ORDER='USER.COZ.SAMPJCL'
//*
//RUNCOZ EXEC PROC=COZPROC,ARGS='-LI user@linux1.myco.com'
//COZCFG DD *
ssh-options=-oStrictHostKeyChecking=no
server-env-PASSWD_DSN=//HLQ.PASSWD(SITE1)  ❶
server-env-SSH_ASKPASS=/usr/local/coz/bin/read_passwd_dsn.sh
server-env-DISPLAY=none
//STDIN DD *
uname -a
env
//
```

❶ The member //HLQ.PASSWD(SITE1) contains a single line with the password starting in the first column and *without* line numbers.

4.2 Running with an OpenSSH keypair

Note: The JCL discussed below is included in the Co:Z toolkit samples as member RUNLNCH

```
//COZUSERC   JOB (),'DOVETAIL',MSGCLASS=H,NOTIFY=&SYSUID,CLASS=A
//PROCLIB JCLLIB ORDER='COZUSER.COZ.SAMPJCL'
//*
//RUNCOZ EXEC PROC=COZPROC,ARGS='cozuser@linux1.myco.com'
//COZCFG DD *
//STDIN DD *
uname -a
env
//
```

4.3 Running with a RACF Digital Certificate

Note: The JCL discussed below is included in the Co:Z toolkit samples as member `RUNLNCHK`

```
//COZUSERC   JOB (),'DOVETAIL',MSGCLASS=H,NOTIFY=&SYSUID,CLASS=A
//PROCLIB JCLLIB ORDER='COZUSER.COZ.SAMPJCL'
//*
//RUNCOZ EXEC PROC=COZPROC,ARGS='cozuser@linux1.myco.com'
//COZCFG DD *
saf-cert=MY-RING
//STDIN DD *
uname -a
//
```

5. Co:Z Launcher Examples

This chapter contains common examples ("recipes") for using the Co:Z Launcher. These examples assume that you have installed and configured the Co:Z toolkit on your z/OS and target systems. Additionally, these examples rely on Dataset Pipes commands. See the *Co:Z Dataset Pipes User's Guide* for the command reference.

5.1 Execute commands on a target server

This is a simple example of how to use the Co:Z Launcher to run commands on a remote Linux server.

```
//COZCB1  JOB (),'COZ'
//STEP1   EXEC PROC=COZPROC,
//        ARGS='myuid@linux1.myco.com'
//STDIN   DD *
# This is input to the remote shell
echo "We are running on: " `uname -sr`
//
```

- The userid and hostname (`myuid@linux1.myco.com`) are given as a parameter to the COZPROC stored procedure, but all other configuration options are taken from the installation defaults.
- The Co:Z Launcher will start an SSH connection to the remote server as user `"myuid"`.
- Since SSH is unable in a batch job to prompt for a password, it will use a private key associated with the current z/OS user to login to the target server.
- The default program to launch on the target server is the user's "default shell", which happens to be "bash".
- Input to the remote shell is redirected from the job's //STDIN DD.
- Output from the remote shell is redirected to //STDOUT DD and //STDERR DD in the launching jobstep. By default these are defined in COZPROC to go to SYSOUT spool files.
- Output from the remote shell is redirected to //STDOUT DD and //STDERR DD in the launching jobstep. By default these are defined in COZPROC to go to SYSOUT spool files.

In this example, the following output will be written to the `//STDOUT DD`:

```
We are running on:  Linux 2.6.15-27-k7
```

The exit code from the remote program (bash) will be adopted as the return code for the batch job step; in this case: "0". Log messages from the Co:Z Toolkit are written to `//SYSOUT DD`:

```
```

```
fromdsn(DD:STDIN)[N]: 2 records/160 bytes read; 75 bytes written in 0 milliseconds.
todsn(DD:STDERR)[N]: 0 bytes read; 0 records/0 bytes written in 0.072 seconds (0.000 Byte
todsn(DD:STDOUT)[N]: 39 bytes read; 1 records/38 bytes written in 0.074 seconds (527.027
CoZLauncher[N]: myuid@linux1.myco.com target command '<default shell>' ended with RC=0
```

5.2 Launch remote shell that reads a PDS member

In this example we use the Co:Z Launcher to send commands to a target Linux server which reads a PDS member from the launching z/OS system.

```
//COZCB2  JOB (),'COZ'
//STEP1   EXEC PROC=COZPROC,
//        ARGS='myuid@linux1.myco.com'
//STDIN   DD *
fromdsn '//sys1.maclib(acb)' | grep BLKSIZE
//
```

- Input to the remote shell is redirected from the job step's //STDIN DD, which in this example has a single line.
- The `fromdsn` command, *running on the target server*, establishes a connection with the launching z/OS job. This connection is used to read a PDS member.
- The single quotes are required so that the Linux shell does not interpret the parentheses as meta characters.
- The `fromdsn` command converts the records in the dataset to a stream of bytes that is written to stdout. By default the data will be converted to a text file using the target platform's codepage and line separator.
- The data is piped ('|') by the shell into the Unix `grep` command which writes matching lines to stdout.
- Output from the remote shell is redirected to //STDOUT DD and //STDERR DD in the launching jobstep. By default, these are defined in COZPROC to go to SYSOUT spool files.

In this example, the following output will be written to the //STDOUT DD:

```
&BLKSIZE=0,&LRECL=0,&BUFSP=0,                                  -00001600
BLKSIZE=&BLKSIZE,LRECL=&LRECL,                                 -01700000
BLKSIZE=&BLKSIZE,LRECL=&LRECL,                                 -02406800
```

5.3 Offload PGP encryption of MVS Datasets

In this example we use the Co:Z Launcher to send commands to a Linux server which reads data from an `//INPUT DD` in the launching job step and writes PGP encrypted output data to `//OUTPUT DD`.

```
//COZCB3  JOB (),'COZ'
//STEP1   EXEC PROC=COZPROC,
//        ARGS='myuid@linux1.myco.com'
//STDIN   DD *
fromdsn -l rdw -k //DD:INPUT   \
  | gpg -r key-1 --batch --output=- --encrypt=- \
  | todsn -b //DD:OUTPUT
/*
//INPUT DD DISP=SHR,DSN=KIRK.CLEARTEXT.DATA
//OUTPUT DD DSN=KIRK.ENCRYPT,DISP=(NEW,PASS),
//          SPACE=(CYL,(1,1),RLSE),
//          DCB=(RECFM=U,BLKSIZE=4096)
```

- Input to the remote shell is redirected from the job step's `//STDIN DD`, which in this example contain three commands chained together with Unix pipes.
- The `fromdsn` command, *running on the target server*, establishes a connection with the launching z/OS job. This connection is used to read from the //INPUT DD. The `-l rdw` option is used so that 4 byte RDWs are used as record separators. This option also disables any default codepage translation. The `-k` option disables any trimming of trailing pad (space) characters from the end of records. The result is that the fromdsn simply writes RDW-prefixed records, as-is, to stdout.
- The output from fromdsn is piped into the Linux `gpg` command which PGP-encrypts the data stream. The `"-encrypt=-"` option causes gpg to read input from stdin (the output pipe from fromdsn).
- The `"output=-"` option causes gpg to write its encrypted output to stdout, which is piped ('|') into a `todsn` command.
- The `todsn` command, running on the Linux server, tunnels back into the launching jobstep and writes the encrypted data stream to `DD:OUTPUT`, which in the example goes to new cataloged MVS dataset. The `"-b"` option causes fromdsn to write the records in binary, with no record separators, in effect filling each record to its maximum size, which is set by the DD card in this case to be 4096 bytes.
- Output log messages from the Co:Z Launcher, the Co:Z Agent (running on the target server), and the Dataset Pipes utilities fromdsn and todsn are written to `//SYSOUT DD`.

In this example, the following log message will be written:

```
fromdsn(DD:STDIN)[N]: 3 records/240 bytes read; 106 bytes written in 0 milliseconds.
fromdsn(DD:INPUT)[N]: 78 records/6240 bytes read; 6552 bytes written in 0 milliseconds.
todsn(DD:OUTPUT)[N]: 2034 bytes read; 2 records/2034 bytes written in 0.038 seconds (52.2
todsn(DD:STDOUT)[N]: 0 bytes read; 0 records/0 bytes written in 0.708 seconds (0.000 Byte
todsn(DD:STDERR)[N]: 0 bytes read; 0 records/0 bytes written in 0.706 seconds (0.000 Byte
CoZLauncher[N]: myuid@linux1.myco.com target command '<default shell>' ended with RC=0
```

Note that the data is encrypted during transfer automatically by the SSH tunnel used by Co:Z to communicate between the target server and the launching batch job. Also note that the file is never stored on disk on the target server.

Decrypting is just as easy:

```
fromdsn -b  //DD:INPUT   \
 | gpg -r key-1 --batch --output=- --decrypt=- \
 | todsn -l rdw  //DD:OUTPUT
```

5.4 Use a Linux server as a secure gateway to information on the Internet

In this example the Co:Z Launcher is used to run a script on a server which downloads a tab-delimited file from the Internet and converts selected fields to SQL statements which are written to a temporary MVS dataset. A second step in the job runs the DB2 batch SPUFI utility to load the data to a DB2 table.

```
//COZCB4  JOB (),'COZ'
//*********************************************************************
//* STEP1: Launch a remote script to ftp download a tab-delimited
//* text file.  Use selected columns to generate DB2 INSERT statments
//* which are written to an MVS temporary dataset.
//*
//STEP1   EXEC PROC=COZPROC,ARGS='cozcb4@dmz1.myco.com'
//STDIN   DD *
wget -O- ftp://ftp.visi.com/users/juan/ContactingCongress.db.txt |
awk -F "\t" -v sq="'" '{
  if (NR == 1)  #skip header/empty table
    print "DELETE FROM CONGRESS;"
  else {
    print "INSERT INTO CONGRESS VALUES("
    print sq $1 sq ", "
    print sq $2 sq ", "
    print sq $4 sq ", "
    print sq $5 sq ", "
    print sq $3 sq
    print ");"
  }
'}
//STDOUT DD DSN=&&SPUFIN,DISP=(NEW,PASS),SPACE=(CYL,(2,1)),
//       DCB=(RECFM=FB,LRECL=80)
//*********************************************************************
/* STEP2: Run DB2 "SPUFI" in batch to execute the insert statements
/* to reload a DB2 table
/*
//STEP2    EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(0,NE)
//SYSTSPRT DD SYSOUT=*
```

```
//SYSPRINT DD SYSOUT=*
//SYSTSIN  DD *
DSN SYSTEM(DBS1)
RUN PROGRAM(DSNTEP2) PLAN(DSNTEP71) LIB('DB2V810.RUNLIB.LOAD')
END
//SYSIN    DD DSN=&&SPUFIN,DISP=(OLD,DELETE)
//
```

- `//STEP1.STDIN DD` contains command input to the remote shell. The `wget` command is used to download a file from the internet using the ftp protocol, piping the output into the `awk` command.
- The `awk` command script, running on the Linux target server, is used to reformat selected columns from the tab-delimited file into SQL INSERT commands, which are written to `stdout`.
- `//STEP1.STDOUT DD` is overridden to point to a temporary MVS dataset, which is passed to STEP2.
- The second step runs the DB2 batch SPUFI utility to execute the SQL, thereby loading the CONGRESS table.

In environments where the z/OS mainframe is not connected to the Internet, the target server may be deployed in a DMZ which is only accessible in one direction from the z/OS host using SSH. Only the target process started by the Co:Z launcher (and its children) have access to redirected I/O resources in the launching Co:Z job step.

This example also demonstrates how open source Linux tools (wget, awk) may be used to access and transform data for use within the z/OS environment.

5.5 Offload processing of SMF data to a Linux system

In this example the Co:Z Launcher is used to offload processing of z/OS SMF data to a Linux system.

```
//USERC4  JOB (),'DOVETAIL',MSGCLASS=H,NOTIFY=&SYSUID
//PROCLIB JCLLIB ORDER='USER.COZ.SAMPJCL'
//*
//*********************************************************************
//*
//* This Co:Z example dumps SMF records to a temporary dataset then
//* remotely processes the records.  The sample program smfp.c
//* reports statistics on the number and types of SMF records
//* processed.  It can easily be modified to perform other processing,
//* or replaced by a SAS program.
//*
//* 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 COZPROC procedure has been installed.
//* 3.) Modify the SMF dataset names for your installation.
//* 4.) Compile the remote SMF processing program "smfp.c" on the
//*     target system:
//*         gcc -o smfp smfp.c.
//*     Ensure that the executable (smfc) is in the path.
```

```
//*
//* When executed, smfp will write a report to stdout similar to the
//* following:
//*
//*                          ------ Length ------
//*Type    Count     Pct    Min     Max      Avg
//*   2        1       0     14      14       14
//*   3        1       0     14      14       14
//*   4        1       0    291     291      291
//*   5        1       0    142     142      142
//*  20        3       0     87      88       87
//*  30      129       2    394    1337      928
//*  42     6304      97    192   19768      467
//*  80        5       0    351     376      360
//*  89        2       0    362    4018     2190
//* 177        1       0     47      47       47
//*
//*6448 SMF records processed
//*
//*******************************************************************
//*
//DUMPSMF  EXEC PGM=IFASMFDP
//SYSPRINT DD   SYSOUT=*
//SMFDATA  DD   DISP=SHR,DSN=SYS1.SMF.DATA
//SMFOUT   DD   SYSOUT=*
//OUTDD    DD   DSN=&&SMFUNLD,DISP=(NEW,PASS),
//         UNIT=SYSDA,SPACE=(CYL,(20,20))
//ADUPRINT DD   SYSOUT=*
//SYSIN    DD   *
    INDD(SMFDATA,OPTIONS(DUMP))
    OUTDD(OUTDD,TYPE(000:255))
/*
//RUNCOZ EXEC PROC=COZPROC,ARGS='user@linux1.myco.com'
//SMFUNLD  DD   DSN=&&SMFUNLD,DISP=(OLD,DELETE,DELETE)
//STDIN    DD   *
fromdsn -b -l rdw //DD:SMFUNLD | smfp
//
```

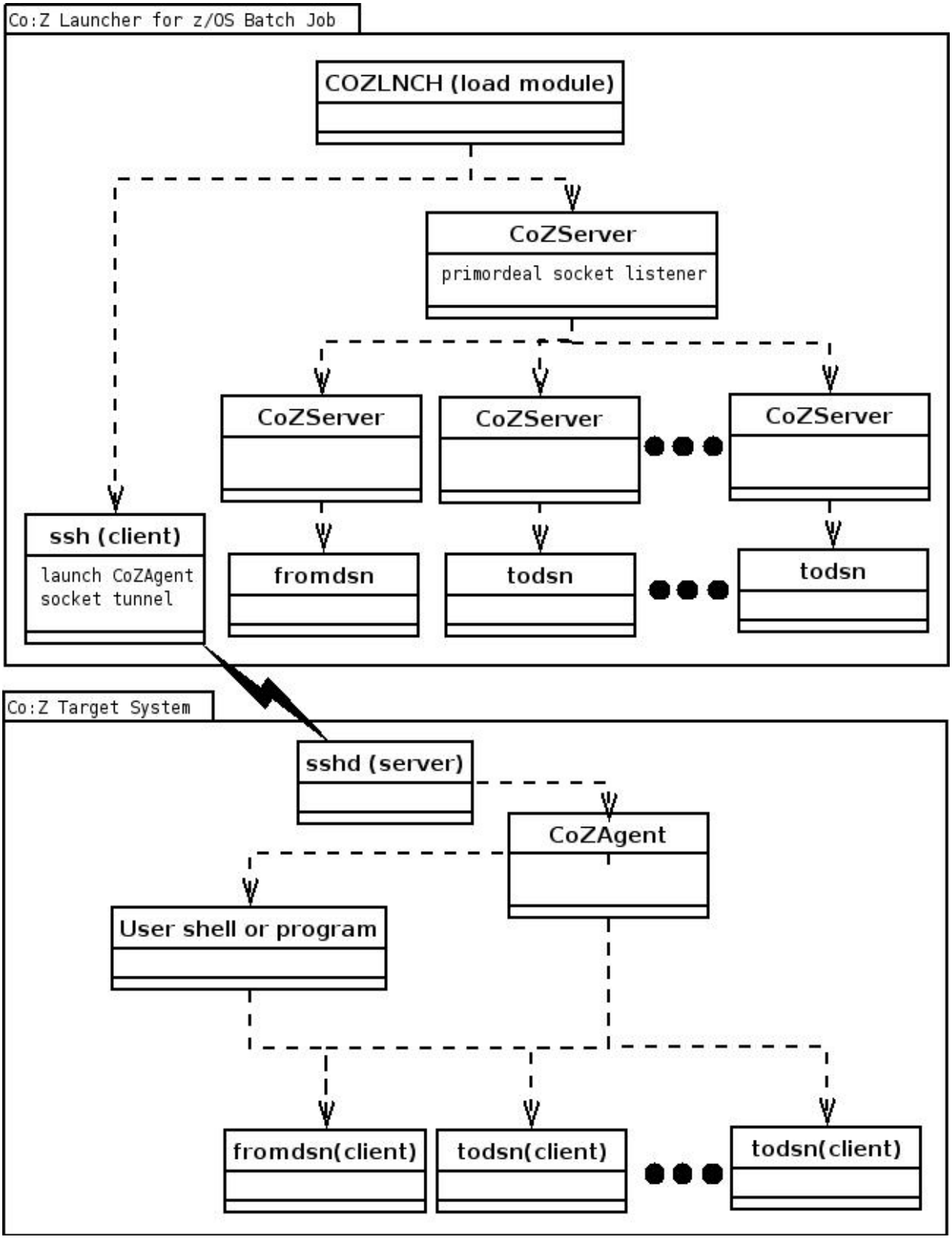
- In most installations, this job would be started as a started task by a IEFU29 SMF dump exit, passing the name of the SMF dataset to dump as an argument to the proc.
- Step DUMPSMF unloads the SMF dataset to a temporary dataset.
- The RUNCOZ step runs the Co:Z Launcher to launch a shell on the target Linux system.
- STDIN input to the Co:Z launcher runs the `fromdsn` command on the Linux target system, which reaches back into the launching z/OS job to read the contents of DD SMFUNLD. The `-b` and `-l rdw` switches cause the SMF data to be read in binary, without translation, and each SMF record to be prefixed by a 4-byte RDW. `fromdsn` supports several RDW layouts, including `-ibmrdw` and `-mfrdw` (for MicroFocus output)
- Source code for the smfp sample program is available on the ***downloads*** page.

6. Problem Determination

This part of the Co:Z User's Guide contains information to help you diagnose problems with the Co:Z Launcher.

6.1 Co:Z Toolkit Component Overview

The following diagram illustrates the processes that occur (on z/OS and the target system) when running the Co:Z Launcher:



6.2 Common Logging/Tracing Options

Several aspects control the logging of messages and trace information in the Co:Z toolkit:

The logging level
: A threshold level: e**M**ergency, **A**lert, **C**ritical, **E**rror, **W**arning, **N**otice, **I**nfo (default), **D**ebug, and **T**race. **F**ine. Each message has a level and will only be logged if that level is at or below the current logging threshold.

The component's logger object
: Each major component (C++ object) in the system will typically have its own logger, which can have its own threshold level set, or can use the default threshold.

The common logging destination
: All logger messages eventually go to the same logging logging destination, which defaults to **stderr**, but a specific file, a user-written routine, or the SYSLOG facility may also be used.

Logging options are set using either the **-L** command-line switch or by by setting the `COZ_LOG` environment variable. In either case, the value of the setting is a list of one or more of the following values:

M|A|C|E|W|N|I|D|T|F
: The default logging threshhold: eMergency, Alert, Critical, Error, Warning, Notice, Info (default), Debug, Trace, Fine.

t
: Prefix log messages with a system timestamp.

e
: Include consumed cpu time in log messages.

f=filepath
: Messages are logged to the given filepath instead of stderr.

s
: Messages are logged to SYSLOG facility instead of stderr.

logname=M|A|C|E|W|N|I|D|T|F
: Set a specific log name to the given threshold

6.3 Enabling Component Logging

The following examples demonstrate how to enable logging for various Co:Z toolkit components:

Set the default logging threshold for the batch launcher job

The following example uses the **-LT** command switch to set the default logging level to "Trace" for all components in the batch launcher job (but not the target system components). The **t** option is also used to prefix all messages with a timestamp.

```
//COZLG1  JOB (),'COZ'
//STEP1   EXEC PROC=COZPROC,
//        ARGS='-LT,t myuid@linux1.myco.com'
//COZCFG DD *
```

```
server-env-COZ_LOG=T,t
//STDIN   DD *
# This is input to the remote shell
echo "We are running on: " `uname -sr`
//
```

Set the default logging threshold for the target system components

The following example uses the **agent-options** property to set the **-L** command line switch to configure CoZAgent logging.

The example also sets the target system environment variable `COZ_LOG` to set logging options for all other target components. Environment variables for the target system can be set using the `target-env-` property prefix in the Co:Z Launcher configuration properties DD (COZCFGD).

```
//COZLG1  JOB (),'COZ'
//STEP1   EXEC PROC=COZPROC,
//        ARGS='myuid@linux1.myco.com'
//COZCFG DD *
agent-options=-LT,t
target-env-COZ_LOG=T,t
//STDIN   DD *
# This is input to the remote shell
echo "We are running on: " `uname -sr`
//
```

Enabling ssh diagnostic messages

It is sometimes useful to increase the verbosity of ssh itself to determine a problem source. To do this for the ssh client (used by the Co:Z Launcher process), add one or more `v` switches to `ssh-options` property in COZCFG:

```
//COZCFG DD *
ssh-options=-vv
//
```

More `v`'s increase the debug level. Note that ssh can produce lots of output.

7. Frequently Asked Questions

The following sections describe the symptoms of several common Co:Z configuration problems.

EDC8127I Connection timed out

If you receive a "EDC8127I Connection timed out" trying to ssh to your Target system, ensure that the ssh daemon (sshd) is started on the target machine. Confirm that you can connect by starting a local ssh session: `ssh -p <port> userid@localhost`. If you can connect locally, ensure that your firewall is not blocking access to your designated ssh port.

cozagent: command not found

Make sure that the Co:Z target toolkit has been downloaded and installed on the target system as described in the installation instructions. By default, the executables, including cozagent are installed in the directory `/opt/dovetail/coz/bin`. If the executables are installed in a different directory, set the `agent-path` property in your JCL to point explicitly to that alternate path.

/usr/bin/cozagent: Permission denied

This is likely due to not properly having the execute bit set on the Co:Z target executables. Locate the directory where they are installed and execute the following: `chmod +x cozagent cozclient fromdsn todsn`

Host key verification failed

Ensure that you have added the target system's host key to `known_hosts` of the userid running the Co:Z Launcher. This is discussed in the installation instructions, but a simple way to do this is to establish an ssh session with the target system from a USS command line and answer "yes" when prompted to add the host:

```
ZOS$ ssh user@68.255.253.94
The authenticity of host '68.255.253.94 (68.255.253.94)' can't be established.
RSA key fingerprint is 09:2c:46:23:56:4e:8f:15:ee:26:5a:12:ec:8d:3a:99.
Are you sure you want to continue connecting (yes/no)? yes
```

Permission denied (publickey,keyboard-interactive).

Usually due to an attempt to connect to a target server with a userid that doesn't have a keypair set up with the calling z/OS system. See the "Configure and test sshd" section in the *Co:Z Target System Toolkits* installation

steps.

command not found for fromdsn or todsn on //STDERR DD

The z/OS Co:Z Launcher uses ssh to first launch the CoZAgent executable at the default path: `/opt/dovetail/coz/bin/cozagent`. CoZAgent then adds its own directory to the `PATH` before invoking the target program or shell.

This is sufficient on most Unix/Linux distributions, but some distributions such as SUSE have default login profiles that reconstruct the PATH variable from scratch, and lose this information when a new login shell is started. In these cases, you will need to update the login profile to include the `/opt/dovetail/coz/bin` directory

Assuming that your default shell is **bash**, here is an example that verifies that an existing PATH variable is not lost by a new login shell:

```
linux$ export PATH=foo:$PATH
linux$ bash --login
(a new shell)
linux$ echo $PATH
(check for the presence of "foo")
linux$ exit
```

If you find that your target distribution has this problem, you will need to update the `/etc/profile` file (or equivalent) to explicitly add the Co:Z binaries directory to the `PATH`.

spawnp(/bin/ssh) - EDC5157I An internal error has occurred. (errno2=0x0B1B0473)

This is likely due to /bin/ssh on z/OS not having the proper file attributes.

Verify that the setuid attribute ("s" bit) is **not** set for either the user or group and that the executable it is **not** APF authorized. Finally, the executable should be allowed to execute in the same address space as the caller. The following output shows the expected settings. If your settings are different, they will need to be corrected.

```
$ ls -al /bin/ssh
-rwxr-xr-x 2 XXXXXX YYYYYY 1531904 Mar 8 2007 /bin/ssh

$ extattr /bin/ssh
/bin/ssh
APF authorized = NO
Program controlled = NO
Shared address space = YES
Shared library = NO
```

Appendix A. 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 A.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 A.3, "OpenSSH SSH_ASKPASS authentication"*.
- Conventional OpenSSH keypairs: *Section A.2, "OpenSSH keypair authentication"*.
- RACF Digital Certificates: *Section A.4, "RACF Digital Certificate authentication"*.

A.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.

A.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*.

A.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.

A.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 B. 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 B.1. Miscellaneous options

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 C. 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.

```
Co:Z 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 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; (d) develop an alternative to the Software that is based on or
derived from, in whole or in part, the Software or Documentation; (e) remove or
obscure any copyright, trademark or other proprietary rights notices or
designations on the Software, the Documentation or any copies thereof; or (f)
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.

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 to perform 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), AND 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 the Software or 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 and waives any objections to the exclusive personal jurisdiction and venue of such courts, including any objection 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 in interest 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.
```

Appendix D. References

D.1 z/OS OpenSSH

Using remote todsn and fromdsn clients requires that *z/OS OpenSSH* or *IBM Ported Tools OpenSSH* be available and configured on z/OS. 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.

D.2 Using the z/OS Unix Shell

The Dataset Pipes **todsn** and **fromdsn** commands may be used from any of the following z/OS Unix shell environments:

- The TSO "OMVS" command
- The **BPXBATCH** utility, running under MVS batch or TSO

  *Note:*The BPXBATCH enhancement **OA11699** significantly improves its usability.
- The z/OS Unix Shell under a telnet or ssh console.

For more information on z/OS Unix, see:

- *z/OS Unix System Services home*
- *z/OS Unix System Services User's Guide*

D.3 The z/OS C library fopen() routine

The Dataset Pipes utilties open MVS datasets in "record mode" using the z/OS C library fopen() routine. For example:

```
fopen( name, mode);
```

where:

name
: either `"//'fully.qualified.dsn'"` or `"//dd:ddname"` depending on whether **BPXWDYN** allocation keywords were used (*Section D.4, "The z/OS BPXWDYN dynamic allocation service"*).

mode

- `"rb,type=record,noseek"` - if reading (fromdsn)
- `"wb,type=record,noseek"` - if writing (todsn)
- `"ab,type=record,noseek"` - if appending (todsn -a)

Additional open mode options may be specifed by using the `-o` option.

The Dataset Pipes utilites read and write records using the z/OS C library `fread()` and `fwrite()` routines. For more information on the capabilities of record-mode dataset processing with the z/OS C library, see:

- *IBM z/OS C++ home*
- *z/OS V1R12 XL C/C++ Run-Time Library Reference*
- *z/OS V1R12 XL C/C++ Programming Guide*. See Ch. 10 "Performing OS I/O operations.

D.4 The z/OS BPXWDYN dynamic allocation service

The Dataset Pipes utilities allow for flexible allocation of MVS Datasets through use of the **BPXWDYN** text-based allocation service. If you specify allocation keywords, either with the `-x` option, or by using the `allocKeywords` option, then a new system-assigned DDNAME will be allocated with BPXWDYN and that DDNAME will be opened with *Section D.3, "The z/OS C library fopen() routine"*fopen().

You may use any allocation keywords defined by BPXWDYN, except the following:

- `DA(), DSN(), FI(), DD(), MSG(), or REUSE()` (automatically supplied)
- `PATH() , PATHDISP(), PATHMODE(), PATHOPTS(), PATHPERM()`
- `RTDDN, RTDSN, RTVOL` (only works if called from REXX)
- `SYNTAX`

For more information on using BPXWDYN allocation keywords, see:

- *z/OS V1R12 Using REXX and z/OS UNIX System Services*

D.5 The z/OS Unicode Translation Services

The Dataset Pipes utilities rely on the *z/OS Unicode Conversion Service* when possible, for codepage/character set translation.

This subsystem provides hardware-assisted high-performance codepage conversions services. This is the same service used by later versions of z/OS DB2, so many shops already have it configured in their environments. For z/OS 1.6 and later, the service is configured by default, with a starter set of codepage (CCSID) mappings.

For more information on configuring and customizing this subsystem:

- *z/OS V1R12 Unicode Services User's Guide and Reference*

When Unicode Conversion Services are not available, Dataset Pipes falls back to **iconv** for codepage translation