

Co:Z Co-Processing Toolkit for z/OS

Co:Z Launcher and Dataset Pipes User's Guide

Steve Goetze
Kirk Wolf

V 2.0.0 Edition

Published December, 2011

Copyright © 2011 Dovetailed Technologies, LLC

Table of Contents

1. Introduction	1
1.1. Co:Z Dataset Pipes Features	1
1.2. Co:Z Launcher Features	2
2. Co:Z Launcher Installation	3
2.1. Configuring the dspipes subsystem (Optional)	3
2.2. Windows Target System Installation	3
Install Cygwin and OpenSSH on Windows	4
Configure and test sshd	4
Install Co:Z target executables	6
2.3. Unix/Linux/Posix Target System Installation	7
Configure and test sshd	7
Install Co:Z target executables	7
3. Co:Z Launcher Configuration	9
3.1. Co:Z Launcher Properties	9
Required Properties	9
Optional Properties	10
3.2. Console communication	11
Commands directed to the CoZAgent process	12
4. Running the Co:Z Launcher	13
4.1. Running with SSH_ASKPASS Authentication	13
4.2. Running with an OpenSSH keypair	13
4.3. Running with a RACF Digital Certificate	14
5. Co:Z Cookbook	15
5.1. Execute commands on a target server	15
5.2. Launch remote shell that reads a PDS member	16
5.3. Offload PGP encryption of MVS Datasets	16
5.4. Use a Linux server as a secure gateway to information on the Internet	18
5.5. Offload processing of SMF data to a Linux system	19
6. Running Dataset Pipes	21
6.1. Running Dataset Pipes with the openssh client	21
6.2. Running Dataset Pipes with the PuTTY ssh client	22
7. Dataset Pipes Examples	25
7.1. Copy an HFS or zFS file to an MVS dataset	25
7.2. Copy to an MVS dataset, overriding target DCB attributes	25
7.3. Copy to an MVS dataset, truncating long lines	25
7.4. Copy to a PDS member	26
7.5. Specifying dataset names	26
7.6. Copying user input to the end of an exiting dataset	26
7.7. Copy an MVS dataset (PDS member) to an HFS file	26
7.8. Copy an MVS dataset using DISP=SHR	27
7.9. Copy one MVS dataset to another	27
7.10. Copy one MVS dataset to another using the same attributes	27
7.11. Copy one MVS non-text dataset to another	27
7.12. Copy an ASCII HFS file to an EBCDIC MVS dataset	28

7.13. Copy an MVS dataset from one z/OS system to another over an SSH connection	28
7.14. From a workstation, download a MVS dataset over an SSH connection	28
7.15. From a workstation, upload an MVS dataset (PDS member) over an SSH connection	29
8. Problem Determination	30
8.1. Co:Z Toolkit Component Overview	30
8.2. Common Logging/Tracing Options	31
8.3. Enabling Component Logging	32
Set the default logging threshold for the batch launcher job	32
Set the default logging threshold for the target system components	33
Enabling ssh diagnostic messages	33
9. Frequently Asked Questions	34
A. Command Reference	36
catsearch	37
cozclient	39
dsn_profile	42
fromdsn	44
lookupccsid	48
lsjes	49
pdsdir	52
safauth	53
saf-ssh-agent	54
showtrtab	56
todsn	59
wto	64
zsym	66
B. Client Authentication Mechanisms	67
B.1. Interactive password authentication	67
B.2. OpenSSH keypair authentication	67
B.3. OpenSSH SSH_ASKPASS authentication	68
B.4. RACF Digital Certificate authentication	68
C. Setting up a test OpenSSH system on z/OS	71
D. Compiling the Co:Z target system sources	73
E. License	74
F. References	79
F.1. IBM Ported Tools for z/OS (SSH)	79
F.2. Using the z/OS Unix Shell	79
F.3. The z/OS C library fopen() routine	79
F.4. The z/OS BPXWDYN dynamic allocation service	80
F.5. The z/OS Unicode Translation Services	80

1. Introduction

Co:Z Dataset Pipes are utilities that convert datasets to/from files. These commands may be used locally or over an ssh connection.

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 Dataset Pipes Features

Co:Z Dataset Pipes can be used in one of three modes:

1. A z/OS unix process accesses local MVS datasets

The Dataset Pipes for z/OS commands, **fromdsn** and **todsn** allow for flexible conversion of record oriented MVS datasets to byte-stream unix pipes.

2. A z/OS jobstep launches a remote process on a target system

The Co:Z Launcher starts a shell process on a distributed system, redirecting its input and output to traditional z/OS datasets or spool files.

The Dataset Pipes client commands can be used by the remote process to reach back into the launching jobstep to access MVS datasets.

The target may be another z/OS system with Co:Z installed.

3. A remote client initiates a connection to z/OS

A Unix, Windows or remote z/OS system can use the Dataset Pipes client commands to initiate an SSH connection to a z/OS server. In this mode, the Dataset Pipes SSH server subsystem (**dspipes**) is used to access MVS Datasets, much like the SSH **sftp-server** subsystem is used to access HFS/zFS files.

Features:

- Pipe input to an MVS dataset (**todsn**)
- Pipe output from an MVS dataset (**fromdsn**)
- Remote execution over an SSH connection
- Supports any MVS dataset which can be opened in sequential, record mode by the `fopen()` C-library routine. This includes:
 - MVS sequential datasets (QSAM, BSAM)
 - PDS and PDSE members
 - VSAM files (processed in sequential mode)

- SYSOUT datasets, including the MVS internal reader
- Supports text or binary conversion via flexible line-termination rules:
 - Cr, Lf/Newline, CrLf, Cr and/or Lf, RDW, none, user-defined-string
- Supports flexible record padding / overflow rules:
 - wrap, flow, truncate, error
- Codepage translation via high-performance z/OS conversion services
- Can specify additional `fopen()` options and dynamic allocation keywords
 - keywords supported by **BPXWDYN** can be used to customize dataset allocation
 - allows for SYSOUT, writers or MVS internal reader
- User and/or system profile can be used to automatically supply conversion options based dataset name matching.

1.2 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



Important

Before proceeding, ensure that the Co:Z Toolkit for z/OS has been successfully installed according to the instructions provided in the document "Co:Z Toolkit Installation and Release Notes" at <http://www.dovetail.com/docs/cozinstall/index.html>. Be sure to make note of the installation directory.

In order to use the Co:Z Launcher (and remote Dataset Pipes clients), the Co:Z Target System Toolkit must be installed on the remote systems that you have identified. You do not need to install Co:Z on a remote system in order to use Dataset Pipes locally.

2.1 Configuring the dspipes subsystem (Optional)

To run Dataset Pipes commands initiated by a remote client, a subsystem must be configured in your z/OS OpenSSH server.¹ This subsystem does not need to be defined if you only want to use the Co:Z Launcher component of the toolkit.

This is done by updating the `sshd_config` file, typically located at `/etc/ssh/sshd_config`.²

Find the line "Subsystem" which defines the `sftp` subsystem. Immediately following the `sftp` line add this:

```
Subsystem dspipes /usr/lpp/coz/bin/dspipes
```

(where `/usr/lpp/coz` is the directory where Co:Z Toolkit is installed).

2.2 Windows Target System Installation

The distribution .zip file for Co:Z includes pre-built binaries for 32-bit Windows systems. The Windows machine must also have OpenSSH installed, which is available as part of the free [Cygwin](#) environment.

Note: Exercise caution when editing text files in the Cygwin distribution, especially shell scripts. Make sure that you use an editor that recognizes and preserves the unix line end characters. Wordpad will work in a pinch, but Notepad will not. If you are comfortable with Unix editors, you can include the vim (vi) package when you install Cygwin.

Install Cygwin and OpenSSH on Windows

¹SSH user subsystems are, like all SSH remote commands, executed in a process under the authenticated client userid, so normal z/OS user security determines what resources can be accessed.

²It is sometimes convenient to set up a *test* OpenSSH server where this subsystem can be easily added. Instructions for doing this can be found in the Co:Z Installation and Release Notes.

Install Cygwin and OpenSSH on Windows

These instructions supplement the information available on the [Cygwin website](#), and must be run under a Windows user with administrator privileges.



Windows Server Installation

The instructions that follow are for standard (non-server) Windows installations outside a Windows domain. For more information about installing Cygwin in a Windows Server Domain environment see this guide:

<http://www.ibm.com/developerworks/wikis/display/tivoliaddm/Setting+up+a+Cygwin+OpenSSH+Server+for+Windows+Domains+on+a+TADDM+Gateway+Server>

1. Download and excute the Cygwin [setup.exe](#) installation wizard
2. Accept the default wizard selections, except where changes are necessary (e.g. "Select Your Internet Connection")
3. After choosing a Download Site, the available packages are listed. Expand the Net node in the package list and click on the Skip: icon next to the package **openssh**. This will cause the openssh and openssl packages to be selected for installation.
4. (Optional) Expand the "Editors" node in the package list and select the vim package if you would like to edit with vi.
5. Wait for the installation to complete. This may take some time depending on the speed of your internet connection.
6. Open a shell: Start+Programs+Cygwin+Cygwin Bash Shell. **NOTE:** This shell must be run as Administrator.

Configure and test sshd

1. Once Cygwin is installed, setting up sshd is simply the matter of running the script **ssh-host-config** from the shell opened in the previous step. Recommended user responses are included below:

```
win$ ssh-host-config
Generating /etc/ssh_host_key
Generating /etc/ssh_host_rsa_key
Generating /etc/ssh_host_dsa_key
Generating /etc/ssh_config file
Privilege separation is set to yes by default since OpenSSH 3.3.
However, this requires a non-privileged account called 'sshd'.
For more info on privilege separation read
/usr/share/doc/openssh/README.privsep.

Should privilege separation be used? (yes/no) yes
Warning: The following function requires administrator privileges!
Should this script create a local user 'sshd' on this machine? (yes/no) yes
```

```
Generating /etc/sshd_config file
```

Warning: The following functions require administrator privileges!

Do you want to install sshd as service?

(Say "no" if it's already installed as service) (yes/no) **yes**

Which value should the environment variable CYGWIN have when sshd starts? It's recommended to set at least "ntsec" to be able to change user context without password.

Default is "ntsec". CYGWIN=**(Enter)**

The service has been installed under LocalSystem account.

To start the service, call `net start sshd` or `cygrunsrv -S sshd`.

Host configuration finished. Have fun!

More information on setting up OpenSSH under Cygwin are available in the `/usr/share/doc/openssh/README` file under the Cygwin home directory.

Note: If you wish to have **sshd** listen on a port other than the default (22) edit the file `/etc/sshd_config` and change the `Port 22` line to reflect the desired port. With Vista, you will need to change the file permissions to do this as the file is owned by a different user id. Be sure to revert the permissions after editing.

2. Start sshd by running `cygrunsrv`:

```
win$ cygrunsrv -S sshd
win$ ps -eaf
  UID      PID      PPID  TTY      STIME  COMMAND
  sgoetze  2644        1  con    16:28:32 /usr/bin/bash
  SYSTEM   4012        1    ?    16:30:53 /usr/bin/cygrunsrv
  SYSTEM   868       4012    ?    16:30:53 /usr/sbin/sshd
  sgoetze  1664      2644  con    16:30:58 /usr/bin/ps
```

3. Test Cygwin ssh locally:



Note

When you supply the Windows userid, it *must* match the case of the actual id on your Windows system.

```
win$ ssh <userid>@localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
RSA key fingerprint is cc:7c:3d:b5:3e:43:5a:6f:12:e2:1a:af:80:45:ae:fa.
```

```
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
<userid>@localhost's password:

win$ logout
Connection to localhost closed.
```

4. Test Cygwin ssh from z/OS:

Repeat the above test from your z/OS userid to confirm that there are no firewall issues.

```
ZOS$ ssh -p <port> <userid>@windows_host
```

Install Co:Z target executables

1. Download Co:Z for Windows from the [downloads](#) page.
2. From a Cygwin bash shell, create the directory /opt if it doesn't exist.
3. Extract the contents of the distribution .zip file to the /opt directory.
4. Ensure that the files in /opt/dovetail/coz/bin are marked executable:

```
$ cd /opt/dovetail/coz/bin
$ chmod +x cozagent cozclient fromdsn todsn
```

5. Add {CYGWIN_HOME}/opt/dovetail/coz/bin to your Windows PATH environment variable and ensure that {CYGWIN_HOME}/bin is also present.

2.3 Unix/Linux/Posix Target System Installation



Note

These steps are required only if you wish to use *nix as a Target system for the Co:Z Launcher or the Dataset Pipes commands remotely. *You do not need to install Co:Z on a remote system in order to use Co:Z SFTP.*

Configure and test sshd

Most Linux and Unix distributions include [OpenSSH](#). Follow the instructions for your operating system for installing and configuring the OpenSSH server (sshd) on your system.

1. Test logging into ssh locally

```
linux$ ssh <userid>@localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
RSA key fingerprint is cc:7c:3d:b5:3e:43:5a:6f:12:e2:1a:af:80:45:ae:fa.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
<userid>@localhost's password: *****

linux$ logout
Connection to localhost closed.
```

2. Test Linux ssh from z/OS:

Repeat the above test from your z/OS userid to confirm that there are no firewall issues.

```
ZOS$ ssh -p <port> <userid>@linux_host
```

Install Co:Z target executables

Co:Z is distributed as a binary LSB compliant RPM for many linux distributions, including Linux for System Z. If you have an LSB 3.0 compliant distribution, installation is very simple and does not require re-compilation.

If a pre-built binary package is **not** available for your operating system, build and install the required Co:Z binaries on your target server as described in [Appendix D, Compiling the Co:Z target system sources](#).

To install an RPM on an RPM based disto, download the appropriate Co:Z LSB from the [downloads](#) page and issue the following command:

```
$ sudo rpm -i coz-toolkit-n.n-m.rpm
```

It is possible to install an LSB RPM on a Debian based distro that is LSB 3.0+ compliant (e.g. Ubuntu Dapper) as well, but it first needs to be converted to a .deb file via `alien`:

```
$ sudo alien coz-toolkit-n.n-m.rpm
$ sudo dpkg -i coz-toolkit-n.n-n.deb
```

Note: the installation directory must be in the default `PATH` used when logging into `ssh`.

On some some distros, you may need to update `/etc/profile` to add binaries to `PATH` (See [this FAQ entry](#)).

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.

Required Properties

Each installation is required to customize the following properties:

`server-path { /usr/lpp/coz/cozserver }`

The absolute path on the server of the CozServer executable.

`server-ports[-sysid] { none }`

The range of ports reserved for communication between CoZServer and the target system. Each invocation of a CoZLauncher 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 CoZLauncher batch job. If `ssh-tunnel=true`, then the target servers must also ensure that these port are available. 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 CoZLauncher 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
```

Optional Properties

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

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.

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 3.2, "Console communication"](#) for a list of available commands.

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.

`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 configuration 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 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 suppresses 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 B, 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 Cookbook

This chapter contains common examples ("recipes") for using the Co:Z toolkit. These examples assume that you have installed and configured the Co:Z toolkit on your z/OS and target systems.

These examples rely on the Dataset Pipes commands, which are included as part of the Co:Z toolkit.

For questions or to suggest new recipes for this cookbook, please visit the [Dovetailed Technologies z/OS Forum](#)

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,  
// ARGVS='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 Bytes)
todsn(DD:STDOUT)[N]: 39 bytes read; 1 records/38 bytes written in 0.074 seconds (527.027 Bytes)
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 (52.7
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. Running Dataset Pipes

The Dataset Pipes commands can be run independently of Co:Z. They work by default on any z/OS system that has Co:Z installed. These commands can also be run from any system that has the Co:Z toolkit for target systems installed if the **dspipes** SSH subsystem has been configured on your z/OS system (see [this installation step](#)).

6.1 Running Dataset Pipes with the openssh client

1. From your shell, test your ssh connection to z/OS: `ssh userid@zoshost env` This should remotely run the **env** command and display the results in your shell. If this doesn't work, refer to the z/OS SSH manual or [OpenSSH man pages](#) for more information.
 - a. The `ssh -vvv` option can be used to enable a protocol trace which can be helpful in diagnosing SSH connection problems.
 - b. The command: `telnet zoshost 22` can be used to verify that you can connect to your z/OS host over the default SSH port.
2. The `fromdsn -ssh` and `todsn -ssh` client commands do not allow the underlying ssh child process to prompt for passwords.

In order to use the clients, you must create SSH authentication keys:

- a. From your shell, Create SSH2 DSA key pair:

```
$ cd
$ mkdir .ssh
$ chmod 700 .ssh
$ ssh-keygen -t dsa
```

- give the private key a passphrase if you care at all about security!
- save the private key in the default location: `~/.ssh/id_dsa`
- save the public key in the default location: `~/.ssh/id_dsa.pub`

- b. In a z/OS Unix shell:

```
zos$ cd
zos$ mkdir .ssh
zos$ chmod 700 .ssh
```

- c. Upload the DSA *public* key (`~/.ssh/id_dsa.pub`) to the (userhome)/.ssh directory, and copy it (to

the end) of `authorized-keys`. Note that this is a text file, so make sure that it is in EBCDIC after you upload it.

For example (from your z/OS Unix shell):

```
zos$ cd ~/.ssh
zos$ cp id_dsa.pub >> authorized_keys
zos$ chmod 600 authorized_keys
```

- d. Under your shell, start a new shell as a child process of **ssh-agent**, which allows it to use your keypair. Under the new shell, use **ssh-add** to add your private key to the agent:

```
$ ssh-agent $SHELL
$ ssh-add
Need passphrase for /home/uid/.ssh/id_dsa..
Enter passphrase: *****
```

Note: it's also possible to setup **ssh-agent** as a cron/daemon process.

3. The following commands can be used (from a shell running under `ssh-agent`) to test **fromdsn** and **todsn**:

```
fromdsn -ssh mypass userid@zoshost 'sys1.maclib(acb)'
```

Example 6.1 display a PDS member

```
cat /etc/profile | todsn -ssh userid@zoshost -r autoexec.bat
```

Example 6.2 upload a text file to the dataset "USERID.AUTOEXEC.BAT"

6.2 Running Dataset Pipes with the PuTTY ssh client

A Cygwin installation is required for Co:Z under Windows per the installation instructions, but you may use PuTTY as your ssh client in place of Cygwin's openssh client if you wish.

1. Download and install the PuTTY commands.
 - a. [Download the PuTTY SSH client](#) commands. **plink** is the only command absolutely required, but you will probably also find **putty**, **pagent**, **puttygen**, **pscp**, and **psftp** useful.
 - b. Put these commands in a directory in your Windows PATH. Refer to the [Putty docs](#) for more information.

2. From a Windows command prompt, test a **plink** remote z/OS command

```
plink -ssh userid@zoshost env
```

This should remotely run the "env" command and display the results in your Windows shell. If this doesn't work, refer to the z/OS SSH manual or PuTTY documentation for more information.

- The `plink -vvv` option can be used to enable a protocol trace which can be helpful in diagnosing SSH connection problems.

The command `telnet zoshost 22` can be used to verify that you can connect to your z/OS host over the default SSH port.

The following commands can be used to test `fromdsn` and `todsn`:

```
fromdsn -ssh -pw mypass userid@zoshost //sys1.maclib(acb)
```

Example 6.3 display a PDS member

```
copy autoexec.bat con: |
  todsn -ssh -pw mypass userid@zoshost -r //autoexec.bat
```

Example 6.4 upload a text file to the dataset "USERID.AUTOEXEC.BAT"

3. The **fromdsn** and **todsn** client commands do not allow the underlying **plink** child process to prompt for passwords. As shown above, the `plink -pw` option can be used to supply your password on the command line.

It's even better to setup SSH authentication keys so that you don't need to supply a password:

- Create SSH2 DSA key pair *using the [puttygen command](#)*
 - Ask for a "DSA" key
 - Give the private key a passphrase if you care at all about security!
 - Save both the public and private key to two separate files
- In a z/OS Unix shell:

```
zos$ cd
zos$ mkdir .ssh
zos$ chmod 700 .ssh
```

- Upload the DSA *public* key to the (userhome)/.ssh directory, and copy it (to the end) of `authorized-keys`. Note that this is a text file, so make sure that it is in EBCDIC after you upload it.

For example:

```
zos$ cd ~/.ssh
zos$ cp dsa_pub.key >> authorized_keys
```

- d. The `authorized_keys` files must have restricted permissions: `chmod 600 ~/.ssh/authorized_keys`
- e. Under Windows, start **pagent.exe**, and add your *private* key to it. This will prompt you (once) for your private key passphrase. It's nice to configure pagent automatically at startup; there's a command switch that lets you specify the private key to use. Then, when ever you login to Windows, you'll see a prompt from Pagent for your passphrase, once you enter it, pagent will sit happily in your system tray.

Once pagent.exe is running in the background with your (unlocked) private key, you never have to supply a password to fromdsn or todsn (or putty and plink) !!

7. Dataset Pipes Examples

This chapter contains common examples for using Dataset Pipes. These examples assume that you have installed and configured the Co:Z toolkit on your z/OS and target systems, and have properly configured the **dspipes sshd** subsystem.

For questions or to suggest new examples for this chapter, please visit the [Dovetailed Technologies z/OS Forum](#)

7.1 Copy an HFS or zFS file to an MVS dataset

```
cat /home/user/myfile | todsn //MVS1.OUTPUT.DATASET
```

This command can be entered from any z/OS Unix shell (see [Section F.2, “Using the z/OS Unix Shell”](#)). The HFS file is copied to `stdout`, which is piped (|) to `stdin` for the **tods**n command which converts the data to records written to the MVS dataset. The default options for **tods**n are in effect:

- Input lines will be broken on CR, LF, or CRLF.
- If the dataset is new, then its default attributes will be "recfm=VB, lrecl=1028".
- Lines longer than allowed by the dataset will be wrapped onto multiple records.

7.2 Copy to an MVS dataset, overriding target DCB attributes

```
cat /home/user/myfile | todsn -o 'recfm=fb,lrecl=80' //MVS1.DATASET1
```

The `-o` option is used to provide additional options to the `fopen()` API. (see [Section F.3, “The z/OS C library fopen\(\) routine”](#)), which is used by **tods**n to open the output dataset. The base `fopen()` options used by **tods**n to open output datasets is "rb,type=record,noseek". Since fixed length records are called for in this example, **tods**n will pad any short records with spaces. (The pad character can be overridden using the `-p` option).

7.3 Copy to an MVS dataset, truncating long lines

```
cat /home/user/myfile | todsn -w trunc //MVS1.DATASET1
```

The `-w` option is used specify how to handle lines longer than the maximum record length of the target dataset. The default is to wrap long lines to a new record. Specify `trunc` to cause long lines to be truncated, or `error` to cause the command to fail if a long line is encountered.

7.4 Copy to a PDS member

```
cat /home/user/myfile | todsn '//MVS1.MYLIB.DATA(MEMBER1)'
```

The single quotes are required so that the parentheses will not be interpreted as shell meta-characters.

7.5 Specifying dataset names

```
cat /home/user/myfile | todsn //userid.test.data
cat /home/user/myfile | todsn -r //test.data
```

- By default, dataset names are assumed to be fully-qualified.
- The `-r` option can be used to automatically add a prefix of the current userid. Assuming that the current userid is "userid", the two above commands use the same dataset.
- Dataset names are always upper case, but upper or lower case names may be given.
- Dataset names that include PDS member names should be enclosed in single quotes, so that the parentheses will not be interpreted as shell meta characters. Quoting the dataset name does not imply anything more; the `-r` option may still be used to indicate that the userid should be added as a prefix.

7.6 Copying user input to the end of an existing dataset

```
tods -a //userid.test.data
```

- Since the `tods` command gets its input from `stdin`, entering the command without a pipe will cause it to read from the terminal. The user can type input lines, ending it `ctrl-d` which signals an end-of-file.
- The `-a` option changes the base `fopen()` options to "ab,type=record,noseek", which opens the file in append (aka "mod") mode. This option can of course be used with pipes as well.

7.7 Copy an MVS dataset (PDS member) to an HFS file

```
fromdsn '//mvs1.my.lib(member1)' > /home/user/member1
```

The **fromdsn** command reads an MVS dataset and converts it to a stream of bytes written to `stdout`. The above command redirects (>) this output to an HFS file. With the default options for `fromdsn`:

- Trailing pad characters (default is spaces) will be removed from the dataset records

- Linefeeds (EBCDIC "newline") characters will be added to the end of each record
- The single quotes are required to prevent the Unix shell from interpreting the parentheses as meta characters.

7.8 Copy an MVS dataset using DISP=SHR

```
fromdsn -x shr //mvsl.input.dataset > /home/user/mydata
```

The default allocation status used by `fopen()` in "read" mode is `DISP=OLD`. The `-x` option can be used to specify `BPXWDYN` allocation keywords (see [Section F.4, "The z/OS BPXWDYN dynamic allocation service"](#)). In this example, the keyword `shr` is used to specify a allocation status of "share", which allows for multiple jobs to read the same dataset simultaneously.

7.9 Copy one MVS dataset to another

```
fromdsn //mvsl.input.dataset | todsn //mvsl.output.dataset
```

The **fromdsn** reads the input dataset and converts it to a stream of bytes which is piped into the **todsn** command which converts that stream of bytes to the output dataset. If the output dataset is new, then the default attributes of `"recfm=vb,lrecl=1028"`. Existing DCB attributes are used if the output dataset already exists. Default line-termination and wrap rules apply, which fine for text data.

7.10 Copy one MVS dataset to another using the same attributes

```
fromdsn //mvsl.input.dataset |  
todsn -x 'new like(mvsl.input.dataset)' //mvsl.output.dataset
```

The `-x` option is used to specify the "new" and "like" `BPXWDYN` allocation keyword, which copies attributes (DCB, SPACE, etc) from a model dataset to allocate the new output dataset. Newline characters are, by default, used as record delimiters, so this command is only appropriate for text datasets.

7.11 Copy one MVS non-text dataset to another

```
fromdsn -k -l rdw //mvsl.input.dataset |  
todsn -l rdw -x 'new like(mvsl.input.dataset)' //mvsl.output.dataset
```

The `-l rdw` option is used on both the **fromdsn** and **todsn** commands to indicate that four byte record-descriptor-words (RDW) should be used in the piped stream to indicate record boundaries. The `fromdsn -k`

option specifies that pad characters should not be trimmed from the end of records (trimming is the default for fixed-length records).

7.12 Copy an ASCII HFS file to an EBCDIC MVS dataset

```
cat /home/user/ascii.txt | todsn -s iso8859-1 -r //my.dataset
```

- The `-s` option names the source codepage(charset) used to convert the data.
- The `-t` option may be used to specify the target codepage.
- If either `-s` or `-t` is omitted, they default to the current codepage for the process's locale, which is commonly "IBM-1047" (EBCDIC, Latin).
- The arguments to `-s` and `>-t` may also be numeric CCSIDs.
- If the same effective CCSID is specified as both the source and target, then no translation is performed.
- The IBM z/OS Unicode Translation service (see [Section F.5, "The z/OS Unicode Translation Services"](#)), is used for all codepage conversions. Starting with z/OS 1.6, this service is configured and enabled by default, but your environment may need to be customized to include specific codepage that you wish to use. If the requested codepage conversions are not available, then Dataset Pipes will try to fall back and use the `iconv()` C-library routine.

7.13 Copy an MVS dataset from one z/OS system to another over an SSH connection

```
fromdsn -k -l rdw //mvs1.input.dataset |  
todsn -ssh user@zos2.myco.com -l rdw //mvs2.output.dataset
```

- **fromdsn** is run locally to create a stream of RDW-delimited records that is piped into the **todsn** command.
- The `todsn -ssh` option creates an SSH client connection over which it runs a remote `todsn` command on the target system.
- The `-ssh` option requires that the "IBM Ported Tools for z/OS (SSH)" product be installed and configured.
- This example assumes that you have configured SSH authentication keys, since the `todsn` command does not allow for password prompting.

7.14 From a workstation, download a MVS dataset over an SSH connection

```
fromdsn -ssh user@zos2.myco.com //mvsl.input.dataset > c:\mydata\data1.txt
```

- **fromdsn.exe** is a Windows program that creates an SSH connection to a remote z/OS host to remotely run the z/OS fromdsn command.
- On Windows, the `-ssh` option requires that the PuTTY **plink** command be installed and available on the PATH.
- fromdsn is also available in source for building on POSIX / Unix systems as part of the Co:Z target server toolkit
- fromdsn.exe has the same arguments and features as the z/OS **fromdsn** command, with the addition of options for specifying the remote z/OS SSH user@host, and optional arguments to SSH / Putty. See the other examples for features of fromdsn that you may remotely use via fromdsn -ssh.
- The linemode option `-l` defaults to `crlf` for the Windows client, and the by default the source codepage will be the same as the current Windows codepage.
- The output of the fromdsn command is the converted stream of data, which is redirected ('>') to a PC file.
- See [Section 2.2, “Windows Target System Installation”](#) for more information

7.15 From a workstation, upload an MVS dataset (PDS member) over an SSH connection

```
copy c:\upload.txt con: |
  todsn -ssh user@zos.myco.com '//userid.lib.data(mem1)'
```

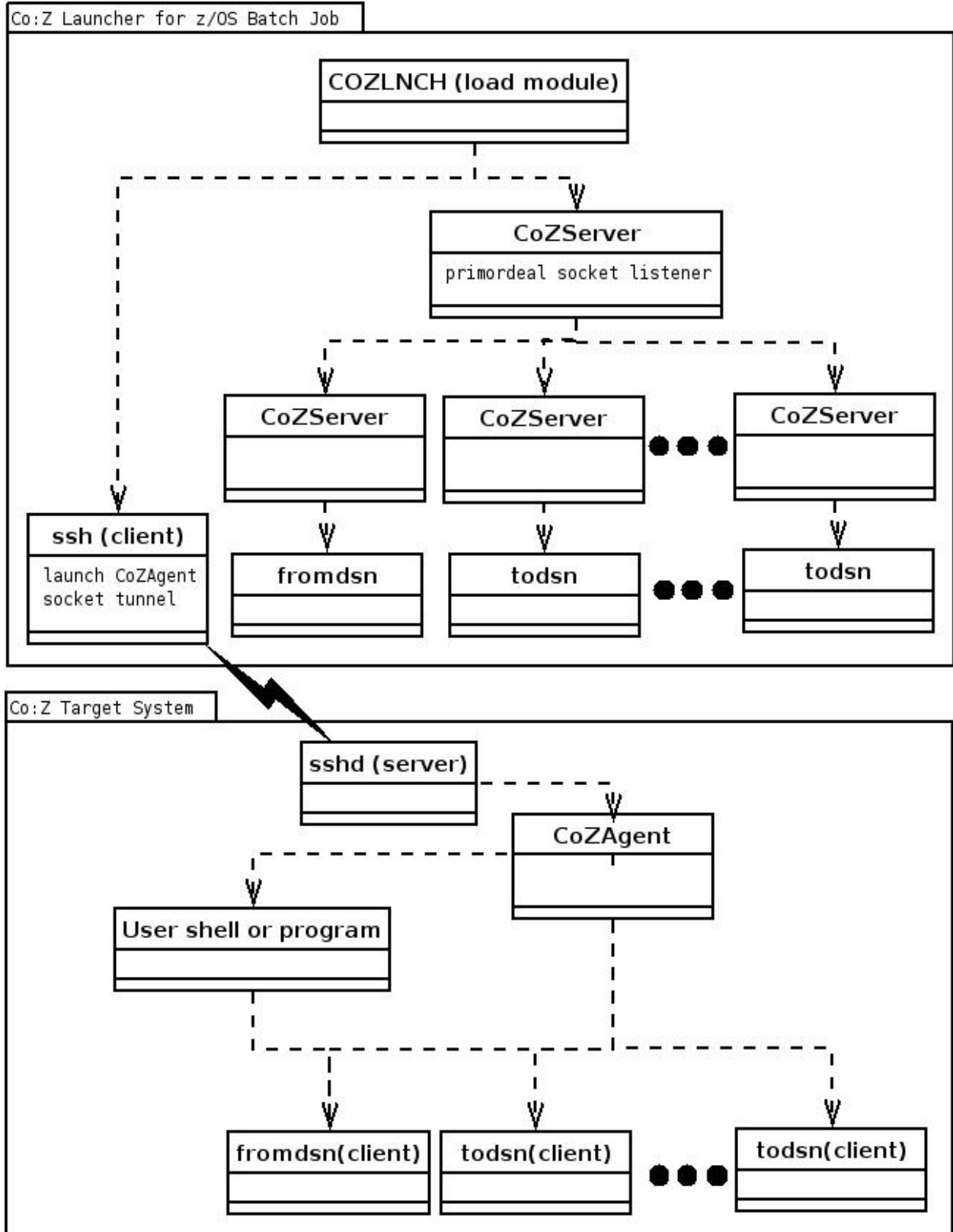
- The Windows copy command is used to pipe (|) the contents of a file into the **tods**n command.
- **tods**n.exe is a Windows executable that creates an SSH connection to a remote z/OS host to remotely run the z/OS todsn command.
- On Windows, the todsn `-ssh` options requires that the PuTTY **plink** command be installed and available on the PATH.
- todsn.exe has the same arguments and features as the z/OS todsn command, with the addition of options for specifying the remote z/OS SSH user@host, and optional arguments to SSH / PuTTY. See the other recipes in this cookbook for features of todsn that you may use remotely with the Windows SSH client.
- See [Section 2.2, “Windows Target System Installation”](#) for more information

8. Problem Determination

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

8.1 Co:Z Toolkit Component Overview

The following diagram illustrates the processes that occur when running the Co:Z toolkit:



8.2 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**Mergergency, **A**lert, **C**ritical, **E**rror, **W**arning, **N**otice, **I**nf (default), **D**ebug, and **T**race. 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

The default logging threshold: eMergergency, Alert, Critical, Error, Warning, Notice, Info (default), Debug, Trace.

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

Set a specific log name to the given threshold

8.3 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 **-LD** command switch to set the default logging level to "Debug" 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 ,
// ARGUMENTS=' -LD,t myuid@linux1.myco.com '
//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'
//STDIN DD *
# This is input to the remote shell
echo "We are running on: " `uname -sr`
/*
//COZCFG DD *
agent-options=-LD,t
target-env-COZ_LOG=D,t
//
```

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.

9. 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 section called "Configure and test sshd"](#) or [the section called "Configure and](#)

test sshd"

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. Command Reference

- *catsearch(1)*
- *cozclient(1)*
- *dsn_profile(5)*
- *fromdsn(1)*
- *lookupccsid(1)*
- *lsjes(1)*
- *pdsdir(1)*
- *safauth(1)*
- *saf-ssh-agent(1)*
- *showtrtab(1)*
- *todsn(1)*
- *wto(1)*
- *zsym(1)*

Name

catsearch — Co:Z utility to list z/OS catalogs

Synopsis

```
catsearch [-l] [-t delim_char] [-m max_entries] [-e entry_types>] filter_key
```

Description

This z/OS Co:Z utility command wraps the Catalog Search Interface (IGGCSI00) and provides a convenient display of information about the Datasets that match the supplied *filter_key*.

The syntax of the *filter_key* and additional documentation can be found in the following IBM publication: *DFSMS: Managing Catalogs - SC26-7409*.

Listing the entire catalog (*filter_key* **) is dis-allowed.

Options

-l

Requests long form information about the listed Datasets. This information includes Volume, last referred date, tracks, used, recfm, lrecl, blocksize, dsorg and Dataset name.

-t

Requests long form information about the listed Datasets in delimited format. If *delim_char* is supplied, it is used as a delimiter, otherwise a tab character (`\t`) is used.

-m *max_entries*

Changes the maximum number of entries that will be returned by catsearch. the default is 2000.

-e *entry_types*

Changes the default entry type filter for catsearch. The default, if not specified, is ABCGHRU. Refer to *z/OS DFSMS Managing Catalogs: Catalog Search Interface* for more information..

Examples

1. This example shows a long listing -l form of a **catsearch**.

```
>catsearch -l user.coz.**
Volume Referred Ext Tracks Used Recfm Lrecl BlkSz Dsorg Dsname
WORK81 2008/09/24 1 30 ? U 0 6144 PO-E USER.COZ.LOADLIB
WORK81 2008/09/24 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 2008/09/24 1 15 4 FB 80 27920 PO USER.COZ.TESTJCL
```

2. This example shows the difference between the single and double asterisk filter key symbols. A single asterisk only lists datasets within the current segment; the double asterisk will span segments.

```
>catsearch -l user.coz.*
```

```

Volume Referred Ext Tracks Used Recfm Lrecl BlkSz Dsorg Dsname
WORK81 2008/09/24 1 30 ? U 0 6144 PO-E USER.COZ.LOADLIB
WORK81 2008/09/24 1 15 4 FB 80 27920 PO USER.COZ.SAMPJCL
WORK81 2008/09/24 1 15 4 FB 80 27920 PO USER.COZ.TESTJCL

>catsearch -l user.coz.**
Volume Referred Ext Tracks Used Recfm Lrecl BlkSz Dsorg Dsname
WORK81 2008/09/24 1 30 ? U 0 6144 PO-E USER.COZ.LOADLIB
WORK81 2008/09/24 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 2008/09/24 1 15 4 FB 80 27920 PO USER.COZ.TESTJCL

```

3. Shows the use of the `-d` switch. Note that only the partial (pseudo directory) is listed for `USER.COZ.TEST`, and that there is no accompanying detailed information. Use of this option can be helpful when dealing with large catalogs.

```

catsearch -dl user.coz.**
Volume Referred Ext Tracks Used Recfm Lrecl BlkSz Dsorg Dsname
WORK81 2008/09/24 1 30 ? U 0 6144 PO-E USER.COZ.LOADLIB
WORK81 2008/09/24 1 15 4 FB 80 27920 PO USER.COZ.SAMPJCL
USER.COZ.TEST
WORK81 2008/09/24 1 15 4 FB 80 27920 PO USER.COZ.TESTJCL

```

Name

cozclient — run a zos command from a remote system via ssh

Synopsis

```

cozclient [OPTION...] command [command-options...]
cozclient -sock [OPTION...] command [command-options...]
cozclient -ssh [ssh-opt...] user@host [OPTION...] command [command-options...]
cozclient -v
cozclient -h

```

Description

The **cozclient** command allows a remote process to execute the z/OS *command* [*command-options*...]. Input (*stdin*) to the command is provided by the remote process and Output (*stdout*) from the command is redirected back to the remote process. Error output (*stderr*) from the command can be routed back to the remote client or to the Co:Z Server's *stderr* stream (if using the *-sock* option).

The z/OS path when executing the command will by default be set to */bin:\$COZ_HOME/bin*.

The **cozclient** command runs in one of the following environments:

- remotely, from a client which was started by Co:Z launcher: *-sock* (default option)
- remotely, from a client-initiated ssh connection: *-ssh* option

Options

-sock

Specifies a remote invocation of **cozclient** from a client environment running under a Co:Z Agent. This is the default. If specified, this must be the first command option.

-ssh [*ssh-options*...] *user@host*

Specifies a remote invocation of **cozclient** using a client-initiated ssh connection to the given z/OS *user@host*. If specified, this must be the first command option.

-h

display help and exit.

-i *stdin_format*

t

stdin sent to the command in text format. Characters are converted from the remote client's codepage to the active z/OS codepage before being sent to the command. This is the default.

b

stdin sent to the command in binary format

n

no stdin is sent to the command

-o stdout_format

t

stdout from the command is sent to the remote client in text format. Characters are converted from the active z/OS codepage to the remote client's codepage. This is the default.

b

stdout from the command is sent to the remote client in binary format

n

stdout from the command is discarded

-e stderr_format

t

stderr from the command is sent to the remote client in text format. Characters are converted from the active z/OS codepage to the remote client's codepage. This is the default.

b

stderr from the command is sent to the remote client in binary format

s

stderr from the command is sent to the Co:Z Server's stderr stream (generally SYSOUT)

-L logging-options

A comma-separated list of options to control logging and tracing:

M | A | C | E | W | N | I | D | T

Logging threshold: eMergency, Alert, Critical, Error, Warning, Notice, Info (default), Debug, Trace.

t

Prefix log messages with a system timestamp

e

Include consumed cpu time in log messages

s

Messages are logged to SYSLOG facility instead of stderr

logname=M | A | C | E | W | N | I | D | T

Set a specific log name to the given threshold

-v

display the current version and exit.

Examples

Remote (via Co:Z Agent) Examples

```
cozclient -in -ot ls -al
```

Run the `ls` command on z/OS. Output is converted to the client codepage and is directed to the remote system's stdout stream.

```
cozclient -in wto "MESSAGE TO CONSOLE"
```

Use the Co:Z toolkit z/OS `wto` command to send a message to the z/OS console.

Remote Client SSH Connection Examples

```
cozclient -ssh user@myzos2.com ls -al
```

Run the `ls` command on z/OS.

```
cat jcl.txt | cozclient -ssh user@myzos2.com submit
```

Submits a job to the internal reader on z/OS. The JCL is contained in the local file `jcl.txt`.

Name

`dsn_profile` — profile information for dataset-name patterns

Synopsis

```
/etc/dsn_profile
~/.dsn_profile
```

Description

todsn and **fromdsn** read dataset-name profile information from `/etc/dsn_profile`, or if present `~/.dsn_profile`. This file contains stanzas of the form:

```
program-name dataset-name-pattern
  keyword value
  keyword value
  ...
```

program-name must start in column 1 of the line and may be either **todsn** or **fromdsn**. Keyword value pairs are read until the start of a new stanza is encountered. Lines starting with '#' and empty lines are interpreted as comments.

`dataset-name-pattern` is a string conforming to the `fnmatch()` C library function pattern language.

The possible keywords and allowed values follow. Keywords are applicable to both **todsn** and **fromdsn** unless noted otherwise. Keywords and values are case-insensitive.

`allocKeywords` | `alloc`

bpxydn dataset allocation options. For a complete list of options, see "Using REXX and z/OS UNIX System Services".

`lineTerminationRule`

`flexible` | `lf` | `cr` | `crlf` | `nl` | `crnl` | `ibmrdrw` | `mfrdrw` | `rdw` | `0xbb[bb..]` | `none`.

`flexible` is only applicable to **todsn**.

`openOptions` | `extraOpenOptions`

Additional mode options to be added to the base options on the `fopen()` call.

`padChar`

the pad character.

`recordOverflowRule`

One of: `error` | `flow` | `trunc` | `wrap`. This keyword is not applicable to **fromdsn**.

`relative`

the `dataset-name` supplied is relative, and the MVS userid will be added.

`sourceCodePage`

the source character set.

`targetCodePage`

the target character set.

`trim`

trailing pad characters are trimmed.

Files

`/etc/dsn_profile`

Contains system wide profile data for **fromdsn** and **todsn**.

`~/ .dsn_profile`

if present, will be read instead, allowing individual users to define their own profile data.

Examples

```
# Force dataset-name containing '.JCL' to be RECFM=FB and LRECL=80
todsn *\e.JCL*
    openOptions recfm=fb,lrecl=80

# Set the codepage and trim option for any dataset name ending with '.ASCII'
fromdsn *\e.ASCII
    targetCodePage ISO8859-1
    trim
```

See Also

`fromdsn(1)`, `todsn(1)`

Name

fromdsn — write the contents of a z/OS dataset to stdout

Synopsis

```
fromdsn [OPTION...] dataset-name
fromdsn -sock [OPTION...] dataset-name
fromdsn -ssh [ssh-opt...] user@host [OPTION...] dataset-name
fromdsn -local dataset-name
fromdsn -v
fromdsn -h
```

Description

The **fromdsn** command reads a z/OS MVS dataset and writes a stream of data to stdout. Lines (if requested) are produced from dataset records based on the options provided.

The **fromdsn** command runs in one of three environments:

- locally (default on z/OS systems)
- remotely, from a client-initiated ssh connection: `-ssh` option
- remotely, from a client which was started by Co:Z launcher: `-sock` (default option on non-z/OS systems)

The user has wide flexibility in choosing:

- How `dataset-name` is to be allocated/opened for writing
- How records are to be created from the incoming source lines
- What character set (codepage) translations are to be performed

`dataset-name` is automatically converted to upper case, and is assumed to be fully qualified unless otherwise specified (see the `-r` option below). If `dataset-name` starts with 'DD:', then it refers to an existing DDNAME.

The **fromdsn** command also supports reading JES spool files using special `dataset-name` syntax:

- `-JES.jobid` - reads the concatenated spool files for a given job.
- `-JES.jobid.dsid` - reads a specific spool file by numerid dsid.
- `-JES.jobid.[stepname[.procstep]ddame` - reads the first spool file in a job that matches a step/procstep/ddname.

Options

-sock

Specifies a remote invocation of **fromdsn** from a client environment running under a Co:Z Agent. This is the default for non-z/OS environments. If specified, this must be the first command option.

-ssh [ssh-options...] user@host

Specifies a remote invocation of **fromdsn** using a client-initiated ssh connection to the given z/OS user@host. If specified, this must be the first command option.

-local

Specifies the use of local z/OS I/O, even if run via CoZLauncher. If specified, this must be the first command option.

-b

binary mode, same as **-l none -p 0x00**.

-h

display help and exit.

-k

keep trailing pad characters in record. The default is to trim if **dataset-name** has fixed length records.

-K

always trim trailing pad characters, even if the dataset contains variable-length records.

-l line-separator

nl | cr | lf | crlf | crnl

follow lines with a newline, carriage return, linefeed, or combination. The characters are taken from the target codepage. The default is **nl**.

ibmrdw

precede lines with a four byte IBM-style RDW, consisting of a two byte network order (big endian) length, followed by two bytes of zeros.

mfrdw

Write a 128 byte MicroFocus standard header prior to output data. Precede each line with a network order (big endian) length. If the maximum record length is < 4095 bytes, the length field is 2 bytes. If the maximum record length is >= 4095 bytes, the length field is 4 bytes. Each line is padded with zeros to the nearest 4 byte boundary.

rdw

precede lines with a four byte network order (big endian) length.

0xbb[bb..]

follow lines with a hex character sequence. The sequence must be between 1 and 8 bytes long.

none

no line separator

-L logging-options

A comma-separated list of options to control logging and tracing:

M | A | C | E | W | N | I | D | T

Logging threshold: eMergency, Alert, Critical, Error, Warning, Notice, Info (default), Debug, Trace.

t

Prefix log messages with a system timestamp

e

Include consumed cpu time in log messages

s

Messages are logged to SYSLOG facility instead of stderr

logname=M | A | C | E | W | N | I | D | T

Set a specific log name to the given threshold

-o fopen-options

additional mode arguments to the z/OS C library fopen() routine. The base mode options used by **fromdsn** to open dataset-name are rb, type=record, noseek". See "z/OS C++ Programming Guide" for details.

-p 0xbb

pad character.

-r

dataset-name will be prefixed with the current z/OS userid.

-s source-codepage

The codepage name or numeric CCSID id of the input dataset. If not specified, then the default z/OS process codepage is used.

-t target-codepage

The codepage name or numeric CCSID id of data written to stdout. If not specified and invoked from a remote client with a line- separator other than 'none', 'ibmrdw', 'mfrdw' or 'rdw', then the default client codepage is used, otherwise the default z/OS code- page is used. Translation is disabled if source-codepage equals target-codepage.

-v

display the current version and exit.

-x bpxwdyn-alloc-keywords

can be specified to provide more precise control over the disposition of dataset-name than the fopen-options. For example, opening a dataset with fopen forces a disposition of 'OLD'. This may not always be desirable in a shared batch environment. The bpxwdyn keywords enable different dispositions to be specified (e.g 'SHR'). If dataset-name is 'DD:name', then this option is ignored. For a complete list of options, see the IBM manual: "Using REXX and z/OS UNIX System Services".

Files

fromdsn may obtain name matched profile information for a dataset from either a per-user profile or a system-wide profile on the z/OS system. For well known dataset-name patterns, profile options can be used to significantly reduce the specification of individual options on the command line. The file format and profile options are described in `dsn_profile(5)`.

Examples

Local z/OS Examples

```
fromdsn mvsl.my.lib(member1) > /home/user/member1
```

Copies an MVS dataset (PDS member) to an HFS/zFS file.

```
fromdsn -x shr mvsl.input.dataset > /home/user/mydata
```

Copies an MVS dataset using DISP=SHR.

```
fromdsn mvsl.input.dataset | todsn mvsl.output.dataset
```

Copies one MVS dataset to another

```
fromdsn -jes.job123 > job.out
```

Copies all output from a job to an HFS/zFS file

```
fromdsn -jes.j333.report.sysprint > report.txt
```

Copies the output from a job's spool file to an HFS/zFS file

Remote Client SSH Connection Examples

```
fromdsn -ssh user@myzos2.com //mvsl.input.dataset > /tmp/data
```

Downloads an MVS dataset over an SSH connection (Unix).

```
fromdsn -ssh user@myzos2.com //mvsl.input.dataset > c:ata.txt
```

Downloads an MVS dataset over an SSH connection (Windows).

```
fromdsn -ssh -p 2222 user@myzos2.com -l rdw -r //binary.dataset >  
/tmp/rdw.bin.data
```

Downloads a MVS dataset over an SSH connection with additional ssh options: (the dataset contains binary records which are prefixed with RDWs)

See Also

`todsn(1)`

Name

lookupccsid — Co:Z utility to return the coded character set identifier (CCSID) associated with a character set

Synopsis

```
lookupccsid codesetName
```

Description

This z/OS Co:Z utility is useful for determining the unicode services CCSID associated with a character set.

This program uses the `__toCcsid()` z/OS C runtime library function to determine the numeric CCSID associated with `codesetName`. If unsuccessful, 0 is returned

Examples

```
/dovetail/coz/bin: > lookupccsid UTF-8  
1208 UTF-8  
  
/dovetail/coz/bin: > lookupccsid ISO8859-1  
819 ISO8859-1
```

Name

lsjes — Co:Z utility to display JES job and spool file status

Synopsis

```
lsjes [-t [delim_char]] [-o userid] [-p jobname-pattern] [-s a|i|o]
```

```
lsjes [-t [delim_char]] -d jobid ...
```

Description

This z/OS Co:Z utility uses the Extended Status Subsystem Interface to query the status of jobs in the primary JES2 or JES3 subsystem.

The first form displays a list, one line per job, all jobs that match optional filter criteria. If no arguments are specified, then all jobs owned by the current userid are displayed.

The second form displays one or more specific jobs, along with their spool files.

Options

-t

Requests output in delimited format. If *delim_char* is supplied, it is used as a delimiter, otherwise a tab character (`\t`) is used. If this option is used, then header lines are not displayed in the listing.

-o *userid*

Filters the job listing to include only jobs whose owner is the given z/OS userid. If this option is omitted, then jobs are filtered using the current userid.

-p *jobname-pattern*

Filters the job listing to include only jobs with a name matching the given pattern. Valid generic pattern characters include '*' and '%'.

-s *a|i|o*

Filters the job listing to include only jobs whose status is either "ACTIVE", "INPUT", or "OUTPUT".

-d

This option indicates the second form of the command (detail mode), in which specific jobs and their spool files are listed. One or more jobids must follow, where each jobid is 2-8 characters that starts with one of the prefixes "J/JO/JOB/T/TS/TSU/S/ST/STC/I/IN/INT" followed by a number.

See Also

The **fromdsn** can be used to read the contents of a job's spool files.

Examples

1. This example lists all jobs owned by the current userid.

```
>lsjes
Jobid   Jobname  Owner   Status   Class  Completion
TSU02611 KIRK     KIRK    OUTPUT   TSU    ABEND=622
JOB02663 KIRKJ1   KIRK    OUTPUT   A      RC=0000
JOB02662 KIRKJ1   KIRK    OUTPUT   A      RC=0000
JOB02661 KIRKJ1   KIRK    OUTPUT   A      RC=0000
JOB02660 KIRKJ1   KIRK    OUTPUT   A      RC=0000
JOB02659 KIRKJ1   KIRK    OUTPUT   A      RC=0000
JOB02462 COZOOM    KIRK    OUTPUT   A      RC=0000
JOB02460 COZOOM    KIRK    OUTPUT   A      RC=0255
```

2. As above, but with delimiters (and without a header).

```
>lsjes -t'|'
TSU02611|KIRK|KIRK|OUTPUT|TSU|ABEND=622
JOB02663|KIRKJ1|KIRK|OUTPUT|A|RC=0000
JOB02662|KIRKJ1|KIRK|OUTPUT|A|RC=0000
JOB02661|KIRKJ1|KIRK|OUTPUT|A|RC=0000
JOB02660|KIRKJ1|KIRK|OUTPUT|A|RC=0000
JOB02659|KIRKJ1|KIRK|OUTPUT|A|RC=0000
JOB02462|COZOOM|KIRK|OUTPUT|A|RC=0000
JOB02460|COZOOM|KIRK|OUTPUT|A|RC=0255
JOB02447|COZOOM|KIRK|OUTPUT|A|RC=0255
JOB02446|COZOOM|KIRK|OUTPUT|A|RC=0255
JOB02334|KIRKSLP|KIRK|OUTPUT|A|RC=0000
JOB02333|KIRKSLP|KIRK|OUTPUT|A|RC=0000
JOB02332|KIRKSLP|KIRK|OUTPUT|A|RC=0000
JOB02331|KIRKSLP|KIRK|OUTPUT|A|RC=0000
JOB02306|KIRKSLP|KIRK|OUTPUT|A|RC=0000
JOB02123|KIRKCB|KIRK|OUTPUT|B|RC=0001
JOB02070|KIRKCT|KIRK|OUTPUT|A|RC=4000
```

3. Tabbed delimiters can be used with the Unix **cut** to select a field:

```
>lsjes -t | cut -f1
TSU02611
JOB02663
JOB02662
JOB02661
JOB02660
JOB02659
JOB02462
JOB02460
JOB02447
JOB02446
JOB02334
JOB02333
JOB02332
JOB02331
JOB02306
JOB02123
JOB02070
```

4. This example lists all active jobs (any owner).

```
>lsjes -o '*' -sa
Jobid   Jobname  Owner   Status   Class  Completion
STC02691 BPXAS    OMVSKERN ACTIVE   STC
STC02689 BPXAS    OMVSKERN ACTIVE   STC
STC02688 BPXAS    OMVSKERN ACTIVE   STC
...
```

5. To list all jobs using a jobname pattern (any owner).

```
>lsjes -o '*' -p 'T*'
Jobid   Jobname  Owner   Status   Class  Completion
STC02556 TCPIP    TCPIP   OUTPUT   STC    RC unknown
STC02579 TCAS     STRTASK OUTPUT   STC    RC unknown
STC02093 TCPIP    TCPIP   OUTPUT   STC    -HELD-
STC02608 TCAS     STRTASK ACTIVE   STC
STC02605 TN3270   TCPIP   ACTIVE   STC
STC02586 TCPIP    TCPIP   ACTIVE   STC
...
```

6. To display the status of a job and list its spool files:

```
>lsjes -d J2333
Jobid   Jobname  Owner   Status   Class  Completion
JOB02333 KIRKSLP  KIRK    OUTPUT   A      RC=0000
      Id  Stepname Procstep DDName   C Owner   Recfm Lrecl Bytes
      002 JES2           JESMSGLG H KIRK    FA     133 1313
      003 JES2           JESJCL   H KIRK    V      136 253
      004 JES2           JESYSMSG H KIRK    VA     137 823
      102 UNIX           SYSOUT   H KIRK    FBA    121 428
```

Name

pdsdir — Co:Z utility to list Partitioned dataset members and their statistics, if available.

Synopsis

```
pdsdir [-n] hlq.dataset.name
```

Description

This z/OS Co:Z utility lists the members of the PDS *hlq.dataset.name* . If statistics are available, they are listed.

Options

-n

Only member names are listed.

Examples

1. This example shows a PDS directory listing.

```
>pdsdir user.coz.sampjcl
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  2008/09/24  17:05:53  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
```

Name

safauth — Co:Z utility to check the current user's authorization to a SAF (RACF) resource.

Synopsis

```
safauth saf-class saf-entity [read | update | control | alter] [volser]
```

Description

This z/OS Co:Z utility is a wrapper for the RACROUTE REQUEST=AUTH macro and can be used to check the current user's access to a given SAF(RACF) resource.

An exit code of zero indicates that the auth check passed; otherwise the non-zero return code from the RACROUTE macro is returned as the exit code.

RACROUTE REQUEST=AUTH requires VOLSER= for CLASS=DATASET, but it is not used for SMS managed datasets. The *volser* option is ignored if CLASS!=DATASET, but if *volser* is not specified and CLASS=DATASET, then *volser* defaults to DUMMY.

Name

saf-ssh-agent — Co:Z utility to enable ssh client authentication via SAF/RACF Digital Certificates

Synopsis

```
saf-ssh-agent -x [-f export_file] keyring[:label]
saf-ssh-agent -b asnl_file keyring[:label]
saf-ssh-agent -c keyring[:label] command [command_args...]
```

Description

This z/OS Co:Z utility is similar in function to the OpenSSH **ssh-agent**, but rather than automatically authenticating the ssh client with ssh keys, it provides for authentication with SAF/RACF Digital Certificates.

keyring[:label] is the keyring (and optional certificate label) to use.

Options

-x

extract the public key from a SAF/RACF Digital Certificate in OpenSSH format.

-f export_file

The file to export the OpenSSH format key. If this option is omitted, the key will be written to `stdout`.

-b asnl-file

extract the public key (in binary ASN1 format) to a file. This option is used for diagnostic purposes.

-c

run *command* as a child process after initializing **saf-ssh-agent**. This enables *command* to authenticate with the supplied *keyring[:label]*. Generally, this option is used to run **ssh** as a child process, allowing it to take advantage of SAF RACDCERT authentication.

Examples

1. This example shows how to extract an OpenSSH public key from a SAF/RACF Digital Certificate. In this case, the key is written to `stdout`.

```
/dovetail/coz/bin: > saf-ssh-agent -x MY-RING

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGDVoW8HzKQYIfVqOZpEHgPLLfUkqg68fyBc
XTDUpFyQiIoKWRh1rHHa4DlQxa80lMPzr+VvyzvJrgzXI00Vp9A09yLgr4XxtrkrfTY3no jT
35y3bZqZXTEfCX5atN8yaORfkXZeY14H+o jdQK3yWHDlqOMTS11Cj4/9w67JNTXXw== CN=
Stephen Goetze,OU=Development,O=Dovetailed Technologies,C=US
```

1. This example shows how to run ssh as a child process to execute the **who** command on the remote system linux.com. The ssh client will authenticate via the SAF RACDCERT contained in MY-RING.

```
/dovetail/coz/bin: > saf-ssh-agent -c MY-RING ssh myid@linux.com who  
myid  tty7          2009-12-29 06:15 (:0)  
myid  pts/0          2009-12-29 11:23 (:0.0)  
myid  pts/1          2010-01-08 11:43 (:0.0)
```

Name

showtrtab — Co:Z utility to display a translation table

Synopsis

```
showtrtab [-L logging_options] [-s source_codepage] [-t target_codepage] [-q technique_string]
```

Description

This z/OS Co:Z utility command will show the translate table associated with a source and target codeset. It first attempts to use unicode services, but will fall back to `iconv()` if needed.

If a table cannot be built, the command will display error information that may be useful in determining the problem.

This utility only supports SBCS -> SBCS and SBCS -> MBCS. MBCS -> SBCS tables are not supported.

To get detailed information, the logging option `-LTranslator=T` can be used.

Options

`-L` logging-options

A comma-separated list of options to control logging and tracing:

M | A | C | E | W | N | I | D | T

Logging threshold: eMergency, Alert, Critical, Error, Warning, Notice, Info (default), Debug, Trace.

t

Prefix log messages with a system timestamp

e

Include consumed cpu time in log messages

s

Messages are logged to SYSLOG facility instead of stderr

logname=M | A | C | E | W | N | I | D | T

Set a specific log name to the given threshold

`-s` source-codepage

The source codepage name. If not specified, then the default z/OS process codepage is used. At least one of `-s` or `-t` is required.

`-t` target-codepage

The target codepage name. If not specified, then the default z/OS process codepage is used. At least one of `-s` or `-t` is required.

-q technique-string

The Unicode Services conversion technique(s) to accept. This is a string of one or more of the following technique characters:

C

Customized Subset

E

Enforced Subset

L

Language Environment Behavior

M

Modified Language Environment Behavior

R

Roundtrip

If more than one character is specified, the first available matching technique is used - therefore the order is significant.

When falling back to `iconv()` this list is ignored

Examples

1. This example shows a Translate table from a source code page of ISO8859-1 to a target codepage which is the current z/OS process' default

```

/dovetail/coz104/bin: > showtrtab -s ISO8859-1

00:  00 01 02 03   37 2D 2E 2F   16 05 15 0B   0C 0D 0E 0F
10:  10 11 12 13   3C 3D 3E 3F   18 19 3F 27   1C 1D 1E 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:  20 21 22 23   24 25 06 17   28 29 2A 2B   2C 09 0A 1B
90:  30 31 1A 33   34 35 36 08   38 39 3A 3B   04 14 3E FF
A0:  41 AA 4A B1   9F B2 6A B5   BB B4 9A 8A   B0 CA AF BC
B0:  90 8F EA FA   BE A0 B6 B3   9D DA 9B 8B   B7 B8 B9 AB
C0:  64 65 62 66   63 67 9E 68   74 71 72 73   78 75 76 77
D0:  AC 69 ED EE   EB EF EC BF   80 FD FE FB   FC BA AE 59
E0:  44 45 42 46   43 47 9C 48   54 51 52 53   58 55 56 57
F0:  8C 49 CD CE   CB CF CC E1   70 DD DE DB   DC 8D 8E DF
    
```

2. This example shows a Translate table from a source code page of ISO8859-2 to a target codepage of IBM-273.

Logging is activated.

```

/dovetail/coz104/bin: > showtrtab -LTranslator=T -s ISO8859-2 -t IBM-273
showtrtab[T]: Translator: -> Translator(ISO8859-2, IBM-273, LMREC)
showtrtab[T]: Translator: -> getCodePage(ISO8859-2)
showtrtab[D]: Translator: Looking for codepage substitution environment
variable: COZ_TRSUB_ISO8859-2
showtrtab[T]: Translator: <- getCodePage()
showtrtab[T]: Translator: -> getCodePage(IBM-273)
showtrtab[D]: Translator: Looking for codepage substitution environment
variable: COZ_TRSUB_IBM-273
showtrtab[T]: Translator: <- getCodePage()
showtrtab[T]: Translator: -> initialize( ISO8859-2->IBM-273, t=LMREC)
showtrtab[T]: Translator: -> getCcsid(ISO8859-2)
showtrtab[T]: Translator: <- getCcsid(912)
showtrtab[T]: Translator: -> getCcsid(IBM-273)
showtrtab[T]: Translator: <- getCcsid(273)
showtrtab[T]: Translator: -> initCunbcprn()
showtrtab[T]: Translator: <- initCunbcprn()
showtrtab[T]: Translator: <- initialize()
showtrtab[T]: Translator: <- Translator()
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 1C 1D 1E 1F
20: 40 4F 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: B5 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 63 EC FC 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 43 BB DC 59 07
80: 20 21 22 23 24 15 06 17 28 29 2A 2B 2C 09 0A 1B
90: 30 31 1A 33 34 35 36 08 38 39 3A 3B 04 14 3E FF
A0: 41 44 46 47 9F 49 52 7C BD 54 57 58 64 CA 66 67
B0: 90 69 70 72 BE 74 77 78 9D 80 8A 8B 8C 8E 8F 9A
C0: 9B 65 62 9C 4A 9E A0 68 AA 71 AB 73 AE 75 76 AF
D0: AC B0 B1 EE EB B2 E0 BF B3 B4 FE B6 5A AD B7 A1
E0: B8 45 42 B9 C0 BA BC 48 CC 51 CD 53 CF 55 56 DA
F0: DB DD DF CE CB EA 6A E1 ED EF DE FA D0 8D FB FD

```

3. Shows an attempt to build a MBCS->SBCS table, and the resulting error.

```

/dovetail/coz104/bin: > showtrtab -s UTF-8 -t IBM-1047
showtrtab[E]: TranslateException: Exception occurred during translation,
RC=4, Reason=12

```

Name

`todsn` — read a stream of data from stdin and write records to a z/OS dataset

Synopsis

```
todsn [OPTION...] dataset-name
todsn -sock [OPTION...] dataset-name
todsn -ssh [ssh-opt...] user@host [OPTION...] dataset-name
todsn -local [OPTION...] dataset-name
todsn -v
todsn -h
```

Description

The **todsn** command writes records to `dataset-name` using a stream of data read from stdin. Dataset records are created from the input stream based on the options provided.

The **todsn** command runs in one of three environments:

- locally (default on z/OS systems)
- remotely, from a client-initiated ssh connection: `-ssh` option
- remotely, from a client which was started by Co:Z launcher: `-sock` (default option on non-z/OS systems)

The user has wide flexibility in choosing:

- How `dataset-name` is to be allocated/opened for writing
- How records are to be created from the incoming source lines
- What character set (codepage) translations are to be performed

`dataset-name` is automatically converted to upper case, and is assumed to be fully qualified unless otherwise specified (see the `-r` option below). If `dataset-name` starts with 'DD:', then it refers to an existing DDNAME.

Options

`-sock`

Specifies a remote invocation of **todsn** from a client environment running under a Co:Z Agent. This is the default for non-z/OS environments. If specified, this must be the first command option.

`-ssh [ssh-options...] user@host`

Specifies a remote invocation of **todsn** using a client-initiated ssh connection to the given z/OS user@host. If specified, this must be the first command option.

`-local`

Specifies the use of local z/OS I/O, even if run via CoZLauncher. If specified, this must be the first command option.

-a
open `dataset-name` in append/mod mode. This option changes the base `fopen()` options to `ab,type=record,noseek`.

-b
binary flow mode, same as `-l none -p 0x00 -w flow`.

-h
display help and exit.

-k
keep trailing pad characters in record. The default is to trim if `dataset-name` has fixed length records.

-K
always trim trailing pad characters.

-l line-separator
`flexible | cr | lf | crlf | nl | crnl`

source lines are separated by combination of linefeed and/or carriage return characters. The default is 'flexible' which allows for any of the other patterns above. These characters are taken from the source codepage.

`ibmrdw`

source lines are preceeded with a four byte IBM-style RDW, consisting of a two byte network order (big endian) length followed by two bytes of zeros.

`mfrdw`

Source data is preceeded by a 128 byte MicroFocus standard header. Source lines are preceeded with a network order (big endian) length. If the maximum record length is < 4095 bytes, the length field is 2 bytes. If the maximum record length is >= 4095 bytes, the length field is 4 bytes. Each record must be padded with zeros to the nearest 4 byte boundary.

`rdw`

source lines are preceeded with a four byte network order (big endian) length.

`0xbb[bb..]`

source lines are followed with a hex character sequence. The sequence must be between 1 and 8 bytes long.

`none`

source lines do not have separators; source lines are determined by the maximum output record length.

-L logging-options
A comma-separated list of options to control logging and tracing:

`M | A | C | E | W | N | I | D | T`

Logging threshold: eMergency, Alert, Critical, Error, Warning, Notice, Info (default), Debug, Trace.

t

Prefix log messages with a system timestamp

e

Include consumed cpu time in log messages

s

Messages are logged to SYSLOG facility instead of stderr

logname=M|A|C|E|W|N|I|D|T

Set a specific log name to the given threshold

-o fopen-options

additional mode arguments to the z/OS C library fopen() routine. The base mode options used by **todsn** to open dataset-name are "wb,type=record,noseek". See "z/OS C++ Programming Guide" for details.

-p 0xbb

pad character. The default is the source codepage space character.

-r

dataset-name will be prefixed with the current z/OS userid.

-s source-codepage

The codepage name or numeric CCSID id of the input data. If not specified and invoked from a remote client with a line-separator other than 'none', 'ibmrdw', 'mfrdw' or 'rdw', then the default client codepage is used, otherwise the default z/OS codepage is used.

-t target-codepage

The codepage name or numeric CCSID id of output dataset. If not specified, then the default z/OS process codepage is used. Translation is disabled if source-codepage equals target-code- page.

-v

display the current version and exit.

-w wrap-options

error

an error is returned if the source line is longer than the maximum record length.

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.

trunc

source lines longer than the maximum record length are truncated

wrap

source lines longer than the maximum record length are broken into multiple records. The default is 'wrap'.

-x bpxwdyn-alloc-keywords

can be specified to provide more precise control over dataset allocation than the fopen-options. These allocation options allow `dataset-name` to be created with specific space and disposition parameters, or allow `dataset-name` to be created like an already existing dataset. If `dataset-name` is 'DD:name', then this option is ignored. For a complete list of options, see the IBM manual: "Using REXX and z/OS UNIX System Services".

-z

allow for an empty input stream. If not specified, the default is to exit with an error and not open or write to the output dataset if the input stream is empty.

Files

todsn may obtain name matched profile information for a dataset from either a per-user profile or a system-wide profile on the z/OS system. For well known `dataset-name` patterns, profile options can be used to significantly reduce the specification of individual options on the command line. The file format and profile options are described in `dsn_profile(5)`.

Examples

Local z/OS Examples

```
cat /home/user/myfile | todsn //MVS1.DATASET1
```

Copies an HFS or zFS file to an MVS dataset.

```
cat /home/user/myfile | todsn -o 'recfm=fb,lrecl=80' //MVS1.DATASET1
```

Copies to an MVS dataset, overriding target DCB attributes.

```
cat /home/user/myfile | todsn -w trunc //MVS1.DATASET1
```

Copies to an MVS dataset, truncating long lines

```
cat /home/user/myfile | todsn -x shr '//MVS1.MYLIB.DATA(MEMBER1)'
```

Copies to a PDS member, allocating with DISP=SHR.

```
cat /home/user/myfile | todsn -r //test.data
```

Specifies a relative dataset name (HLQ will be added).

```
cat /home/user/ascii.txt | todsn -s iso8859-1 -r //my.dataset
```

Copies an ASCII HFS file to an EBCDIC MVS dataset.

```
cat /home/user/rdw.bin | todsn -l rdw -r //my.dataset
```

Copies a binary HFS file with RDW-prefixed lines to an MVS dataset.

Remote Client SSH Connection Examples

```
cat /tmp/data | todsn -ssh user@myzos2.com -r //my.dataset
```

Uploads an MVS Dataset over an SSH connection (Unix).

```
copy c:ata.txt con: | todsn -ssh user@myzos2.com -r //my.dataset
```

Uploads an MVS Dataset over an SSH connection (Windows).

```
cat /tmp/data | todsn -ssh user@myzos2.com -r '//my.pds(mem1)'
```

Uploads a MVS PDS Member over an SSH connection (Unix).

```
copy c:ata.txt con: | todsn -ssh user@myzos2.com -r '//my.pds(mem1)'
```

Upload an MVS PDS Member over an SSH connection (Windows).

```
cat /tmp/data | todsn -ssh -p 2222 user@myzos2.com -r '//my.pds(mem1)'
```

Uploads an MVS Dataset with additional ssh options.

See Also

fromdsn(1)

Name

wto — Co:Z utility to issue a Write To Operator (WTO) from USS.

Synopsis

```
wto [-r ROUTCDE, . . .] [-d DESC, . . .] message
```

Description

This z/OS Co:Z utility command issues *message* as a write to operator (WTO).

If the ROUTCDE or DESC codes are omitted, the system uses the routing code specified on the ROUTCODE keyword on the DEFAULT statement in the CONSOLxx member of SYS1.PARMLIB.

NOTE: The message will be prefixed by: BPXM023I (userid) unless the userid has access to "BPX.CONSOLE" in the SAF "FACILITY" class.

Messages with embedded spaces must be quoted.

Options

-r ROUTCDE

Specifies the routing code(s) for the message:

1 - Operator Action

2 - Operator Information

3 - Tape Pool

4 - Direct Access Pool

5 - Tape Library

6 - Disk Library

7 - Unit Record Pool

8 - Teleprocessing Control

9 - System Security

10 - System/Error Maintenance

11 - Programmer Information

12 - Emulation

13-128 - See *MVS Programming: Authorized Assembler Services Reference, Volume 4 (SETFRR-WTOR) - SA22-7612*

-d DESCR

Specifies the descriptor(s) for the message:

- 1 - System Failure (*)
- 2 - Immediate Action Required (*)
- 3 - Eventual Action Required (*)
- 4 - System Status (*)
- 5 - Immediate Command Response (*)
- 6 - Job Status (*)
- 7 - Task-Related
- 8 - Out-of-Line
- 9 - Operator's Request
- 10 - Not Defined
- 11 - Critical Eventual Action Required (*)
- 12 - Important Information (*)

(*) Mutually exclusive

Examples

1. This example shows a WTO, using ROUTCDE "Programmer Information" and DESCR "Important Information".

```
>wto -r 11 -d 12 "status message"
```

Name

`zsym` — Co:Z utility to list system symbol values.

Synopsis

```
zsym "&symbol"
```

Description

This z/OS Co:Z utility lists the value of *symbol*. Note that the symbol must be preceded by an ampersand (&) and enclosed in quotes.

Examples

1. Show various system symbol values

```
>zsym "&SYSNAME"  
S0W1  
>zsym "&SYSPLEX"  
SVSCPLEX  
>zsym "&YYMMDD"  
080925
```

Appendix B. Client Authentication Mechanisms

Running the Co:Z Launcher and/or the Co:Z SFTP client 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 Batch jobs.

- Interactive password: *Section B.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 B.3, “OpenSSH SSH ASKPASS authentication”*.
- Conventional OpenSSH keypairs: *Section B.2, “OpenSSH keypair authentication”*.
- RACF Digital Certificates: *Section B.4, “RACF Digital Certificate authentication”*.

B.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 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 shell invocations of Co:Z SFTP `cozsftp`.

B.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: a z/OS shell invoked under telnet, rlogin, or ssh must be used for key generation. Don't attempt to do this under an OMVS shell, since the `ssh` commands are generally not supported under OMVS.

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 dsa
Generating public/private dsa key pair.
```

```

Enter file in which to save the key (/home/<userid>/.ssh/id_dsa): <enter>
Enter passphrase (empty for no passphrase): <enter>
Enter same passphrase again: <enter>
Your identification has been saved in /home/<userid>/.ssh/id_dsa.
Your public key has been saved in /home/<userid>/.ssh/id_dsa.pub.
The key fingerprint is:
dd:ff:00:87:43:11:fa:7b:0d:84:3a:19:3b:7f:5d:2e <userid>@<host>

```

The private key file `id_dsa` 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_dsa.pub authorized_keys
Uploading id_dsa.pub to /home/sgoetze/.ssh/authorized_keys
id_dsa.pub          100% 601    0.6KB/s   00:00
sftp> quit

```

Note: If you are adding public keys from more than one z/OS userid to `authorized_keys`, then you must append each key rather than replacing the file as shown above.

B.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.

A dataset member (e.g.) `//HLQ.PASSWD(SITE1)` must be created that contains a single line with the password starting in the first column and *without* line numbers.

B.4 RACF Digital Certificate authentication

Traditional OpenSSH keypairs and `SSH_ASKPASS` are convenient, but some sites have strict policies about keeping user credentials in a SAF facility. The z/OS Communications Server FTP command can exploit RACF Digital Certificates for authentication and encryption. The Co:Z toolkit provides a similar capability via its `saf-ssh-agent` which can be used in conjunction with a user RACDCERT RSA certificate to provide OpenSSH client authentication.

An existing SAF/RACF Keyring 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 Keyring 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 keyring MY-RING

```
//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
    WITHLABEL('MY-CERT' )

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

/* Connect the certificate to the ring */
RACDCERT ID(COZUSER) CONNECT ( + ❶
    ID(COZUSER) + ❶
    LABEL('MY-CERT' ) +
    RING(MY-RING) +
    DEFAULT + ❸
    USAGE(PERSONAL) )

/* Refresh to activate */
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 Makes this certificate the default in the ring. This allows the user to specify just the keyring 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 keyring and certificate, the user must have READ access to the FACILITY class resources:

- IRR.DIGTCERT.LIST
- IRR.DIGTCERT.LISTRING

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 keyring name with `USERID/`. In order to do this, the administrator must have UPDATE access to the SAF classes listed above.

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 public keys from more than one z/OS userid to `authorized_keys`, then you must append each key rather than replacing the file as shown above.

Appendix C. 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 C.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 [*IBM Ported Tools for z/OS 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 /usr/lpp/coz/bin/dspipes
```

(where /usr/lpp/coz 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 /usr/lpp/coz 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 D. Compiling the Co:Z target system sources

Note: These sources have been built on a variety of POSIX systems, but we offer no guarantees for your particular system. If you have difficulty building, please feel free to contact us regarding our testing and certification schedule.

1. Transfer (in binary) the `coz.v.r.m-src.tar` file included in the Co:Z distribution to your target system.
2. Unpack the source:

```
linux$ mkdir cozbuild
linux$ cd cozbuild
linux$ tar xvf ../coz.v.r.m-src.tar
```

3. Build and install:

```
linux$ ./configure
linux$ make
linux$ make install
```

Note: the installation directory must be in the default `PATH` used when logging into `sshd`.

On some some distros, you may need to update `/etc/profile` to add binaries to `PATH` (See [this FAQ entry](#)).

Appendix E. License

The Co:Z Co-Processing Toolkit, comprised of Co:Z Launcher, Co:Z Dataset Pipes, Co:Z SFTP, Co:Z Batch, and the Co:Z Target System Toolkits 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. The Co:Z Target Systems Toolkit and the Co:Z FTP-SSH Proxy and are also available in source form under the [Apache V2 License](#).

For more information, see our licensing [frequently asked questions](#).

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, and the 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 F. References

F.1 IBM Ported Tools for z/OS (SSH)

Using remote `todsn` and `fromdsn` clients requires that SSH be installed and configured on z/OS. The following manuals can also be consulted for assistance:

- [*IBM Ported Tools for z/OS home*](#)
- [*IBM Ported Tools for z/OS User's Guide*](#)

F.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 User's Guide*](#)

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

The Dataset Pipes utilities 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 F.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 specified by using the `-o` option.

The Dataset Pipes utilities 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 V1R7.0 XL C/C++ Run-Time Library Reference*](#)
- [*z/OS V1R7.0 XL C/C++ Programming Guide*](#). See Ch. 10 "Performing OS I/O operations."

F.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 F.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 V1R7.0 Using REXX and z/OS UNIX System Services*](#)

F.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 V1R7.0 Support for Unicode: Unicode Services*](#)

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