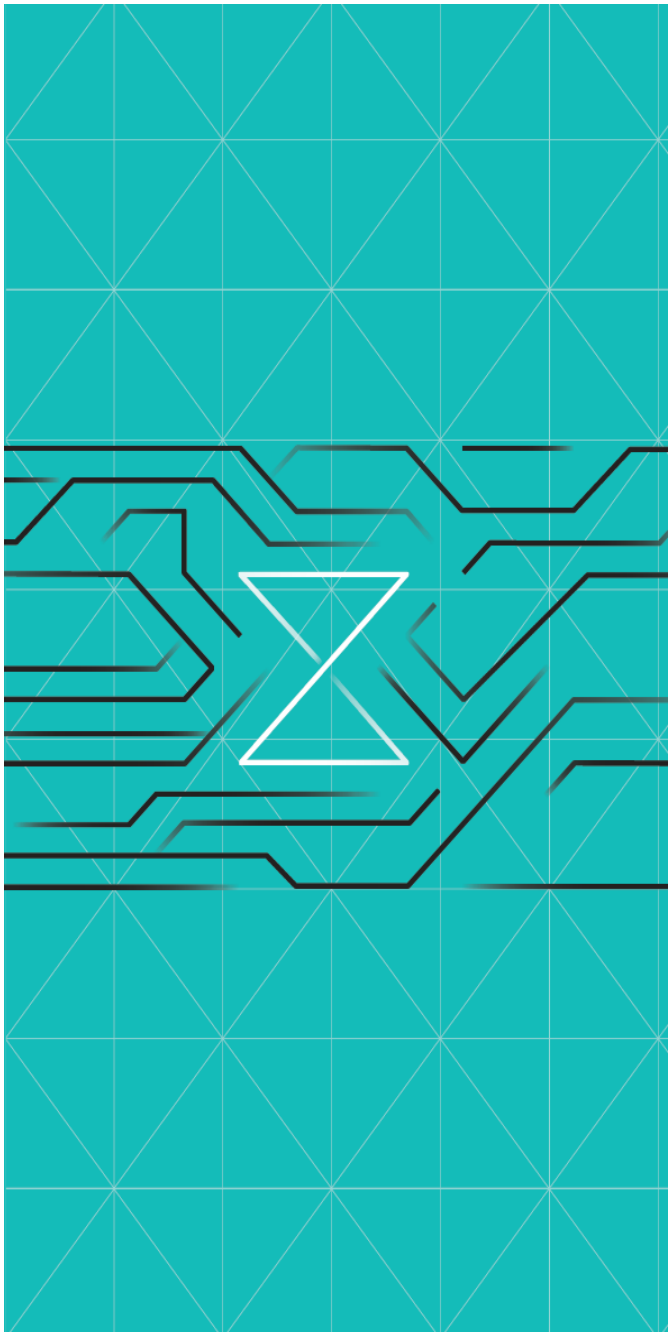


Advanced

250127-1538

ZCL1

Zowe CLI and VSAM

250127-1538

SLINGING RECORDS WITH ZOWE CLI AND VSAM

The challenge builds on your experience with Zowe, shows you more of the options available, and opens up the world of VSAM.

The Challenge

You interact with the mainframe through a series of transactions. You issue a request to view the jobs, another to view data sets, another to issue a command.

Behind the scenes, the open source framework, Zowe, is working to link the capabilities of the mainframe with easy-to-use APIs, commands, and libraries.

Simply put, you can tap into a IBM Z system from just about anywhere, using a wide variety of tools and platforms.

Before You Begin

This challenge will make most sense if you have already completed all of the Fundamentals challenges, as it uses a little bit of everything from there.

Nothing is required, but assumptions will be made about what you know at this stage.

Investment

Steps	Duration
15	120 minutes

ZCL1I250127-1538

1 INSTALL THE ZOWE COMMAND LINE INTERFACE

You been using the Zowe Explorer plugin for VSCode throughout the Fundamentals challenges, but Zowe does much much more, and is responsible for bringing so much more to the mainframe.

Just to be clear, you are installing the Zowe CLI **on your own computer, not on the mainframe** (it is already there!).

Zowe CLI is a **node.js** package, which requires node to be installed and available in your computer operating system.

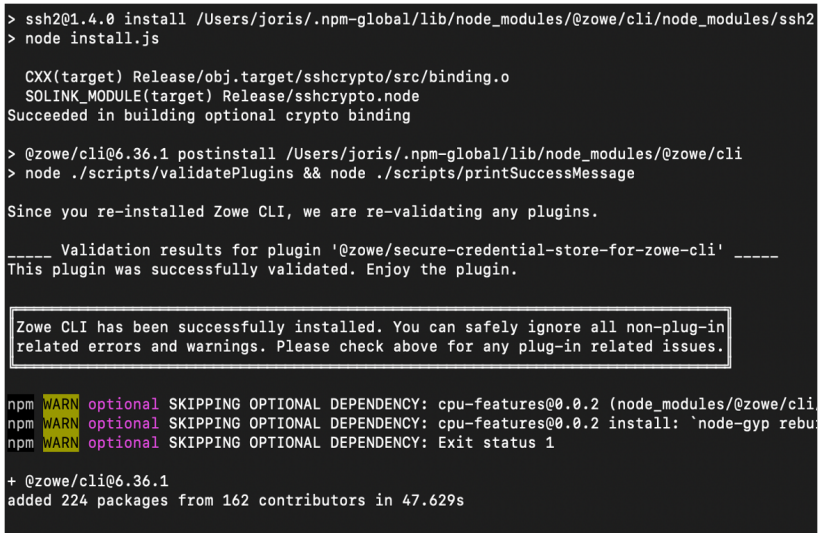
You will use Zowe CLI to interface with Zowe and z/OSMF which is running on the mainframe, but you will drive most of this challenge from your own computer.

Linux users may need to do a bit of exploring to find what works on your specific system distribution, but it should look closer to the MacOS steps, just substituting your correct shell profile file.

2 INSTALL ON MACOS

In order to use node.j packages in the operating system, they need to be loaded into a ***.npm-global*** directory which can be accessed by regular users.

The following steps will set that up, tell **npm** (the Node Package Manager) to use it, and include that in the normal list of places it looks for programs to run.



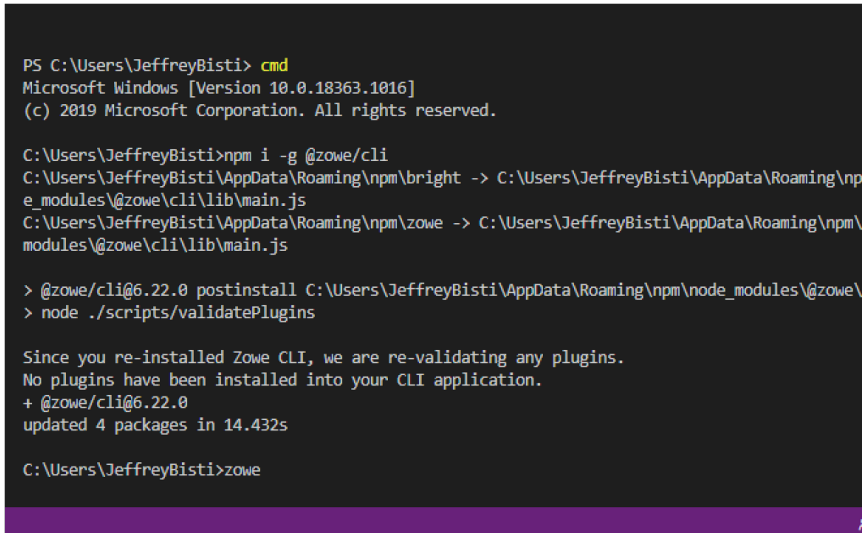
For users of MacOS, these should do the trick.

1. mkdir ~/.npm-global
2. npm config set prefix '~/.npm-global'
3. echo "export PATH=~/.npm-global/bin/:$PATH" >> .zprofile
4. source .zprofile
5. npm i -g @zowe/cli
6. zowe

(Linux users should find the steps work there, too)

3 INSTALL ON MICROSOFT WINDOWS

On Windows, you will find it simpler to first switch to **cmd** from **PowerShell**; then install Zowe CLI using npm.



This should work for most users, though your output may look slightly different than what you see in the screenshot.

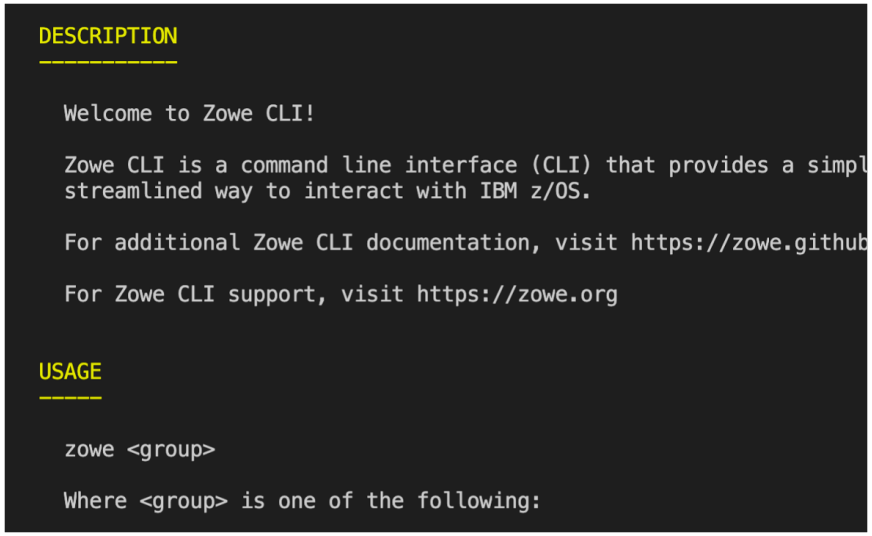
1. Open a Command terminal
2. type `cmd` (this will ensure the shell is Cmd, not PowerShell)
3. npm install @zowe/cli
4. Close the terminal
5. Open a Command terminal
6. type `cmd` (this will ensure the shell is Cmd, not PowerShell)
7. `zowe`

Generally, this should work; however, Windows laptops may have group policies in effect which can limit the ability to instance software, or update the PATH environment setting.

Still stuck? Hop into the forums for guidance

4 A FOUR-LETTER COMMAND

Now that your CLI is all set up, grab yourself a fresh terminal, and type the command `zowe`. Just like that, all by itself.



ZCL1250127-1538

You should get back a description, a listing of command groups, and options. You will be spending a lot of this challenge going through these command groups, and some of them should sound familiar.

If you want to know more than the information in the command details, go to https://zowe.org to learn more.

5 BUILDING PIECE BY PIECE

You've been using the functionality of Zowe to issue commands and do all sorts of things through VSCode. In this challenge, you will use the standalone CLI component to do things in a different way, which can be useful in some situations.

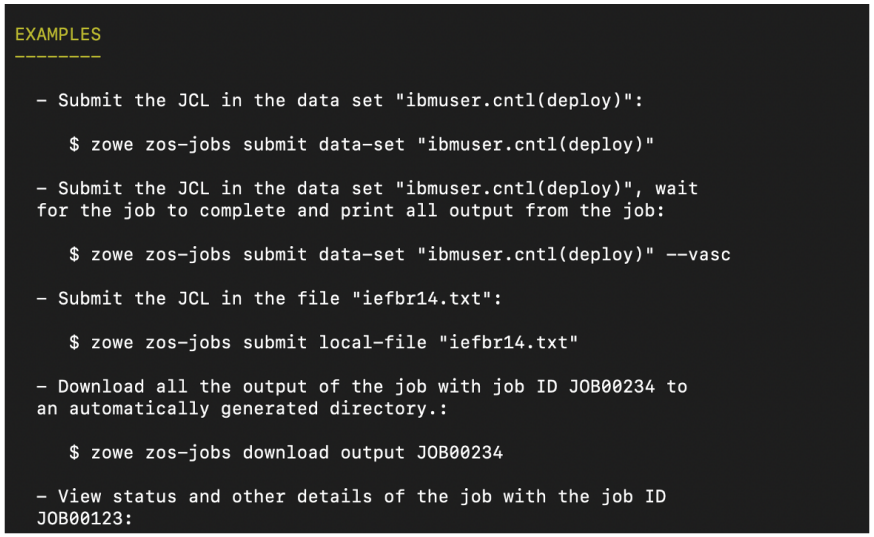
For example, to see what else can be done in the console group, enter the command `zowe console`.

```
USAGE
-----

  zowe zos-console <group>

  Where <group> is one of the following:

GROUPS
------

  collect Collect z/OS console command responses
  issue   Issue z/OS Console Commands

GLOBAL OPTIONS
--------------

  --response-format-json  | --rfj (boolean)

     Produce JSON formatted data from a command

  --help  | -h (boolean)

     Display help text

  --help-examples  (boolean)
```

You can see there is an option to issue commands, as well as collect responses. Those are two "sub-command" groups within console.

6 EXAMPLES ARE GOOD DOC

Use the command `zowe zos-jobs --help-examples` for a nice listing of zowe commands you can use related to z/OS jobs.



```
EXAMPLES
--------

  - Submit the JCL in the data set "ibmuser.cntl(deploy)":

      $ zowe zos-jobs submit data-set "ibmuser.cntl(deploy)"

  - Submit the JCL in the data set "ibmuser.cntl(deploy)", wait
  for the job to complete and print all output from the job:

      $ zowe zos-jobs submit data-set "ibmuser.cntl(deploy)" --vasc

  - Submit the JCL in the file "iefbr14.txt":

      $ zowe zos-jobs submit local-file "iefbr14.txt"

  - Download all the output of the job with job ID JOB00234 to
  an automatically generated directory.:

      $ zowe zos-jobs download output JOB00234

  - View status and other details of the job with the job ID
  JOB00123:
```

The output goes on beyond what is captured in the screenshot above, and there are plenty of variations made available.

Start with the basics but don’t worry, this will get a little more exciting in just a few more steps.

“TELL ME MORE ABOUT ZOWE. IS THIS AN IBM THING OR ...?”

Zowe is an open source project for z/OS, aimed at making the platform more accessible to users who aren’t starting out with years and years of mainframe experience.

The Zowe project contains contributions from individuals as well as companies in the mainframe community. These include the VSCode plugin, a number of APIs, and the Zowe CLI which you are to exploring now.

Zowe is a project of Open Mainframe Project, which is a project managed by the Linux Foundation.

It is *not* an IBM product, though IBM is a contributor and supporter, and continues to advocate for Zowe as a strategic model for bringing new capabilities and users to the mainframe platform.

One of the best ways to get connected to employers and people “in-the-know” is to pay attention to what is happening in these communities and help out whenever you see an opportunity.

ZCL1I250127-1538

7 FORMAT FOR JSON

Make sure you have your VSCode JOBS view active so that you can see your running jobs, and enter the command `zowe zos-jobs list jobs`

You get back a listing of actively running z/OS jobs that you have access to look at. Neat!

Now, issue the same command with `--rfj` (Response Format JSON) after it.

```
Monorail:~ jbisti$ zowe zos-tso issue command "status" --rfj
{
  "success": true,
  "exitCode": 0,
  "message": "",
  "stdout": "IKJ56455I Z99999 LOGON IN PROGRESS AT 08:41:30 ON JULY 14, 2020\nIKJ56951I NO BROADC
Y \n\nIKJ56216I NO JOBS FOUND+\nREADY \n\n",
  "stderr": "",
  "data": {
    "success": true,
    "startResponse": {
      "success": true,
      "zosmfTsoResponse": {
        "servletKey": "Z99999-78-aabeaaaq",
        "queueID": "1507336",
        "sessionID": "0x4E",
        "ver": "0100",
        "tsoData": [
          {
            "TSO MESSAGE": {
              "VERSION": "0100",
              "DATA": "IKJ56455I Z99999 LOGON IN PROGRESS AT 08:41:30 ON JULY 14, 2020"
            }
          },
          {
            "TSO MESSAGE": {
              "VERSION": "0100",
              "DATA": "IKJ56951I NO BROADCAST MESSAGES"
            }
```

Now you get the FULL output, and the output is in JSON format, which can be much more easily interpreted by programs that prefer JSON format.

8 ALLOCATE AND LIST

Take a look at the zowe files command group and use that to “allocate” (create) a **sequential** dataset named **Zxxxxx.ZOWEPS** (use your Z userid, of course)

Next, use another zowe cli command to show the attributes of the dataset you just created. They should look similar to the following:

```
MTM> zowe zos-files list ds Z99999.ZOWEPS -a

  dsname: Z99999.ZOWEPS
  blksz:  6160
  catnm:  MASTERV.CATALOG
  cdate:  2020/08/28
  dev:    3390
  dsorg:  PS
  edate:  ***None***
  extx:   1
  lrecl:  80
  migr:   NO
  mvol:   N
  ovf:    NO
  rdate:  ***None***
  recfm:  FB
  sizex:  15
  spacu:  CYLINDERS
  used:   0
  vol:    VPWRKA
  vols:   VPWRKA
```

In the unlikely event you get a timeout error message, you might want to try adding --responseTimeout 30 to the end of the command, to allow for delays in response.

9 FULLY CUSTOMIZED

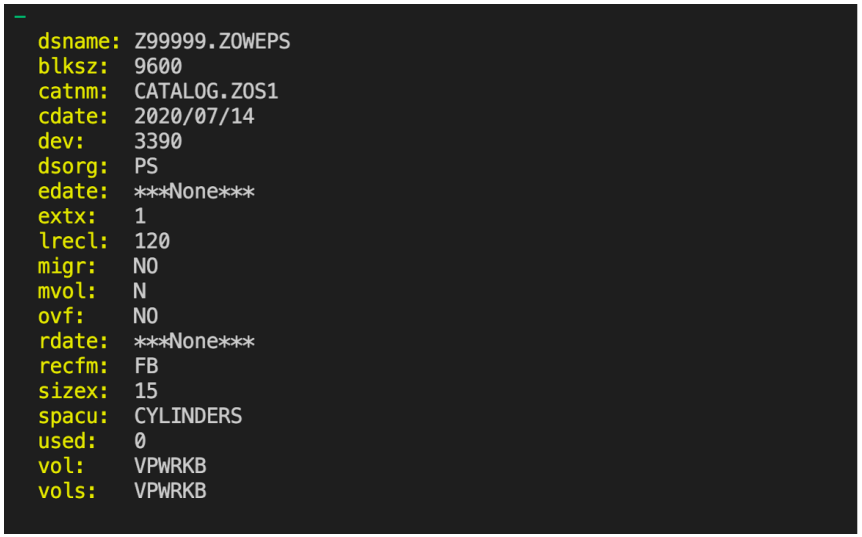
So now you know yet another way of creating and looking at datasets.

The thing is, you made that dataset using a set of default values, and one of the great things about datasets is how customizable they are.

Delete the **ZOWEPS** dataset (with another `zowe files` command) and use the help (or online documentation at https://zowe.org) to re-create that sequential data set with some customized attributes.

The requirement this time is for the Record Length to be 120 instead of the default 80 (there will got some long records to store in the dataset), and a specific block size of 9600.

When you get it, you'll see a different readout for the Block Size (blksz)and Record Length (lrecl) properties, like in the screenshot below.



What about JSON?

Why do we need JSON when the original output made perfect sense to us?

JSON stands for **J**ava**S**cript **O**bject **N**otation, and it's just a way of nesting the attributes of something into an object so it can be fully represented whenever it's accessed.

It is often seen to be a little more lightweight and flexible than another file format with a similar goal you may have heard about, called XML.

In many programming languages, you can simply load in a JSON object, and then use "dot notation" to access the various attributes of that JSON object, saving valuable time when programming, compared with the manual task of writing parsers to extract information from regular output.

ZCL1250127-1538

10 THE KEYS TO THE DATA

```
EXAMPLES
--------

  - Create a VSAM data set named "SOME.DATA.SET.NAME" using
  default values of INDEXED, 840 KB primary storage and 84 KB secon

     $ zowe zos-files create data-set-vsam SOME.DATA.SET.NAME

  - Create a 5 MB LINEAR VSAM data set named
  "SOME.DATA.SET.NAME" with 1 MB of secondary space. Show the prope
  data set when it is created:

     $ zowe zos-files create data-set-vsam SOME.DATA.SET.NAME --dat
secondary-space 1MB --show-attributes

  - Create a VSAM data set named "SOME.DATA.SET.NAME", which is
  retained for 100 days:

     $ zowe zos-files create data-set-vsam SOME.DATA.SET.NAME --ret
```

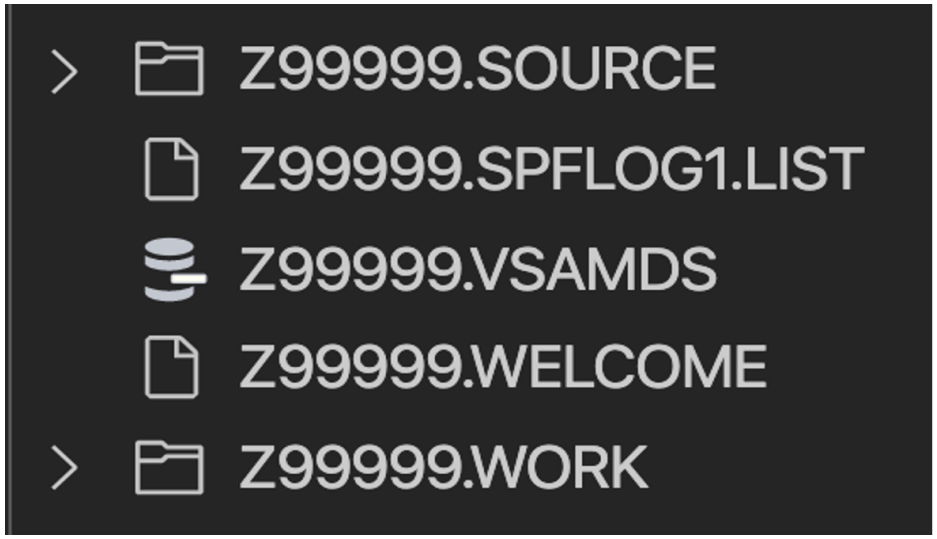
One type of dataset you have seen in the Zowe menus is **VSAM**, and it deserves special attention.

VSAM is not used for things like storing JCL or “Welcome to the Mainframe” messages.

Its time to shine is when an application needs to access records as quickly and efficiently as possible. In fact, without special software to interpret VSAM files, you can’t open them up in a normal editor, but applications happily eat those files right up.

It’s all about efficiency in data access. Read more below.

11 BUILD A VSAM CLUSTER



You’re obviously getting good at allocating data sets. Make a VSAM data set called **Zxxxxx.VSAMDS**.

Refer to the Zowe online help for a guide to the command.

When done, look at its attributes (you know how) and you’ll notice something pretty interesting; it looks like there are THREE data sets here. Plus, if you view it in your DATA SETS list in VSCode, you’ll see a snazzy new icon.

Curious yet?

12 LOAD IT UP

Empty datasets are not very useful - so now you're going to add some records.

You can use the existing sample data in **ZXP.PUBLIC.SAMPDATA**, or you can have some fun and make your own. https://Mockaroo.com has a nice data generator you can try out, though a few notes:

1. The first column (the "keys") must be in (ascending) order
2. Omit any blank records/rows
3. You will need leading zeroes for keys, otherwise VSAM may not see them as being in order when you try to import
4. Make sure this new input data is stored in a zOS sequential dataset or a PDS member.

Download the sample **REPRO** member from the **ZXP.PUBLIC.JCL** dataset to your personal workstation, placing it in the directory you're currently working from.

Note that if you created your own input dataset, you will need to edit the JCL to point to your source dataset name.
Name your REPRO file as *repro.txt*

```
zowe jobs submit local-file "repro.txt"
```

can be used to submit the JCL directly from your machine, through the Zowe CLI.

You'll see a nice little animation, and a job number. Check that job number and make sure it ran smoothly.

```
1IDCAMS  SYSTEM SERVICES
0
      REPRO -
            INFILE(INPUT) -
            OUTDATASET(Z99999.VSAMDS)-
            ERRORLIMIT(6)
0IDC0005I NUMBER OF RECORDS PROCESSED WAS 1000
0IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
0
0IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

Here I am. Allocate me like VSAM

VSAM is complicated, and this little grey box is not going to give you years of experience working with VSAM datasets, but it will tell you that if you want that mainframe job, do all the reading and practicing with VSAM data sets that you can. They are a core component of any big mainframe company.

For now, know that there are four main types of VSAM data sets -

KSDS	key-sequenced data set
ESDS	entry-sequenced data set
RRDS	relative record data set
LDS	linear data set

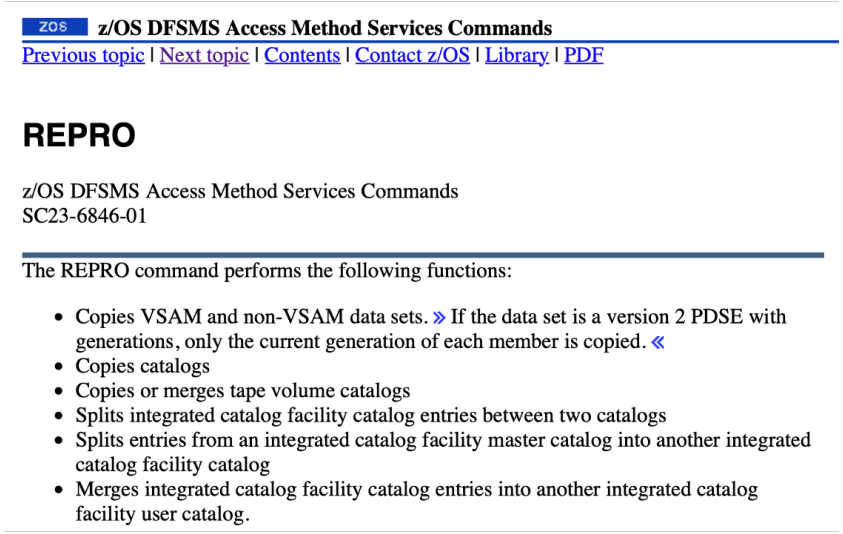
KSDS and ESDS are the most common, and the difference comes down to how each record gets stored and accessed.

KSDS means that you reference a key (like looking up an account number) and getting the information for that account as the record.

ESDS stores data in a sequential order, for data that is likely to be read one after the other in a particular order.

More details available at zOS Concepts

13 LET'S TAKE INVENTORY



z/OS z/OS DFSMS Access Method Services Commands

Previous topic | Next topic | Contents | Contact z/OS | Library | PDF

REPRO

z/OS DFSMS Access Method Services Commands
SC23-6846-01

The REPRO command performs the following functions:

- Copies VSAM and non-VSAM data sets. » If the data set is a version 2 PDSE with generations, only the current generation of each member is copied. «
- Copies catalogs
- Copies or merges tape volume catalogs
- Splits integrated catalog facility catalog entries between two catalogs
- Splits entries from an integrated catalog facility master catalog into another integrated catalog facility catalog
- Merges integrated catalog facility catalog entries into another integrated catalog facility user catalog.

ZCL1250127-1538

Let's talk about what you just did. The JCL runs **IDCAMS**, which is primarily used to manage VSAM datasets. Within IDCAMS, you used the REPRO command to load a sequential set of records into a VSAM-formatted dataset.

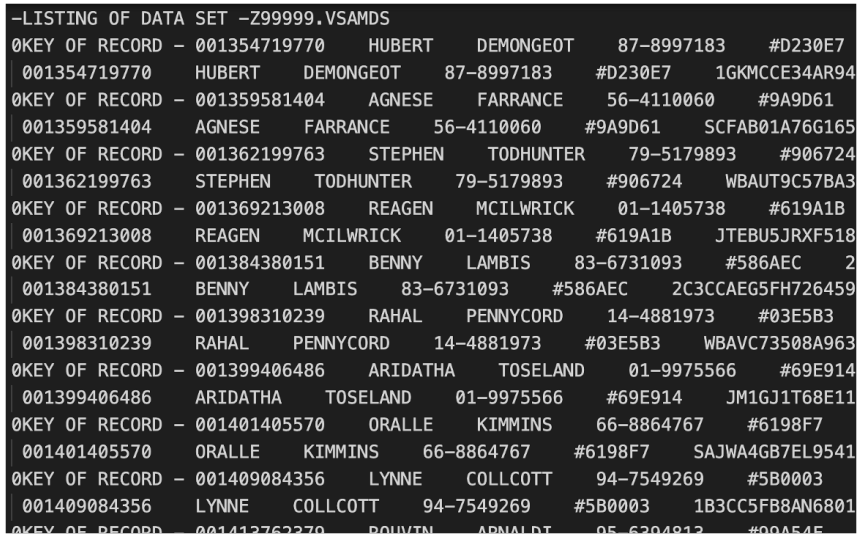
There's a LOT of complexity happening "behind the scenes" that you don't see, but there's plenty of opportunity to control exactly how you want that copy to happen, including cryptographic parameters.

The data is the same, but it is now structured fundamentally different, indexed by key, and able to be referenced much more efficiently by programs (including some written in REXX!)

In reality, this data is not indexed very well, since each line is its own key, but if we dive into the particulars of building a VSAM cluster, you can see how the keys and record size can be specified.

Much more detail in the IDCAMS introduction

14 PRINTING THE RECORDS



There is another very handy IDCAMS command to look at your data - the aptly-named **PRINT** command.

Details for the PRINT command

Check out the example (hint) and pay attention to the CHARACTER parameter (hint hint) to help produce output to look like the above screenshot.

You'll be putting together information from several sources here, so think about what you have, and what you want. You will want to print out that VSAM dataset in character format.

Does that help? Don't be afraid to stop by the forums for some help.

15 MAKE IT COUNT

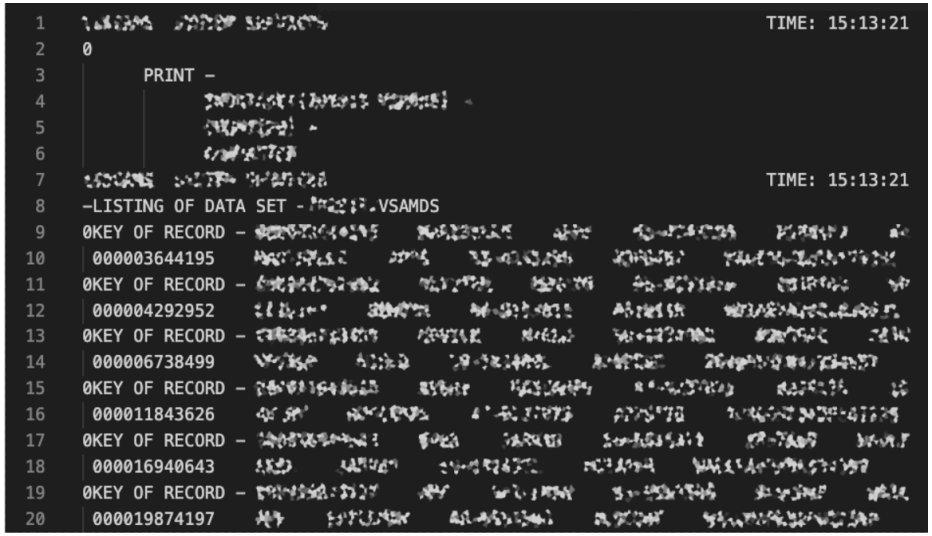
Come for the Zowe CLI, stay for the VSAM and IDCAMS.

To complete this challenge, the validation will check for 3 things:

1. Your Zxxxxx.ZOWEPS sequential data set with the right properties
2. Your Zxxxxx.VSAMDS VSAM data set, loaded with data records
3. The first 20 lines of output from an IDCAMS PRINT command, copy/pasted into a sequential Zxxxxx.OUTPUT.VSAMPRNT data set.

Don’t write the VSAMPRNT data set directly from your JCL; use the joblog SYSPRINT, and copy/paste lines 1-20 of your SYSPRINT.

The checker will look for the header.



Refer to the screenshot above as a (lightly redacted) example. When you have completed the task, submit the checker job **CHKAZCLI**

Nice job - let's recap	Next up ...
You came into this challenge probably not knowing much about **Zowe CLI**, and are leaving knowing not only how to get around in Zowe CLI, but a little bit about VSAM and IDCAMS. You've probably also noticed the level of instruction starting to shift from "here's a command - run this exactly" to "figure this out". Welcome to the big leagues, this is how we roll now	Now that Zowe CLI is in your toolkit, and you have some basic VSAM experience, let's give you some more details on how VSAM works.