ZCLI1

Zowe CLI and VSAM

Slingling records with Zowe CLI and VSAM

15 steps 120 minutes

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 mainframe's capabilities with easy-to-use APIs, commands, and libraries. Simply put, you can tap into a Z system from just about anywhere, using a wide variety of tools and platforms.

BEFORE YOU BEGIN

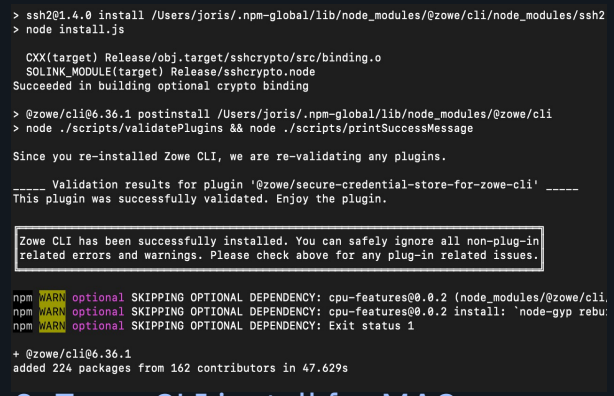
This challenge will make most sense if you've already completed all of the Fundamentals challenges, as it uses a little bit of everything from there. Nothing is required, but we will make assumptions about what you know at this point.

1. Installing Zowe CLI

We've been using the Zowe Explorer plugin for VS Code throughout this contest, but Zowe does much much more, and is responsible for bringing so much more to the mainframe.

To be clear, **you're installing Zowe CLI on your own computer, not on the mainframe**. You'll use Zowe CLI to interface with Zowe and z/OSMF which is running on the mainframe, but you'll be driving most of this challenge from your own computer.

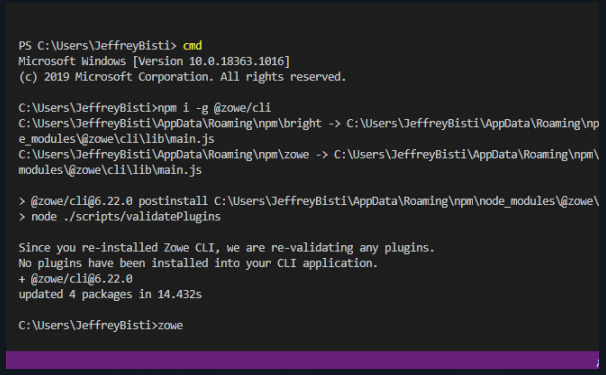
Linux users may need to do a bit of exploring to find what works on your specific system, but it should look closer to the Mac steps, just substituting your correct shell profile file.



2. Zowe CLI install for MAC

In order to use node packages in the operating system, we need to load them into an .npm-global directory which we can be accessed by regular users. These steps will set that up, tell npm (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**

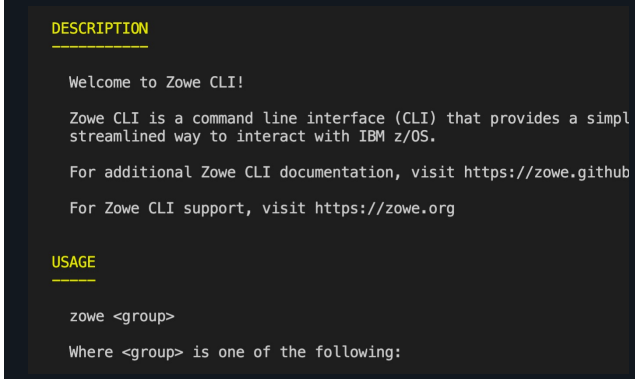


3. NPM setup for Windows

On Windows, we're first going to switch to cmd from PowerShell, then install zowe zli using npm, the Node Package Manager. This should work for most users, though your output may look slightly different than what you see in the screenshot.

1: type **cmd** (this will change the shell to cmd from PowerShell)
1: **npm i -g @zowe/cli**
2: **zowe**

Still stuck? Hop into the forums for guidance

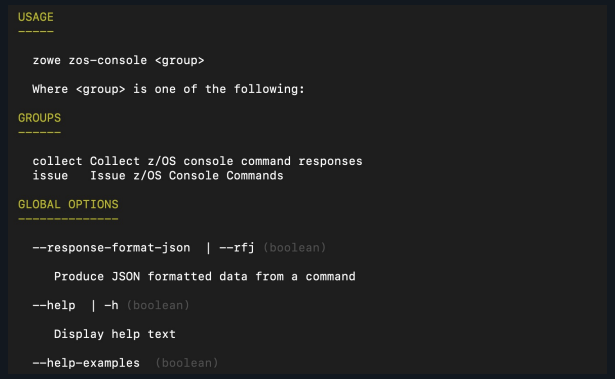


4. A four letter command

Now that we're all set up, grab yourself a fresh terminal, and type the command **zowe**. Just like that, all by itself.

Make sure you follow the profile setup instructions from steps 5-6 of the REXX1 challenge, otherwise this might fail.

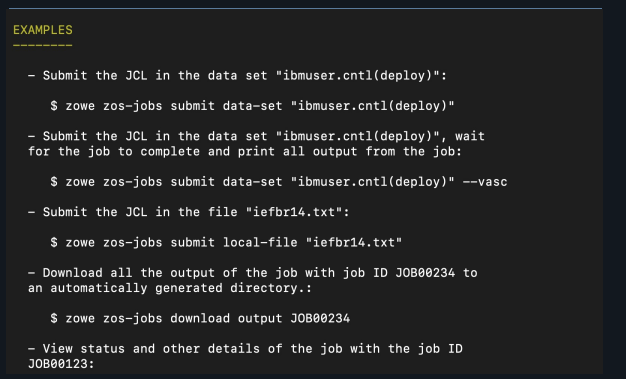
You'll get back a description, a listing of command groups, and options. We'll be spending a lot of this challenge going through these command groups, and a lot of them should sound somewhat familiar. Go to zowe.org to learn more.



5. Building piece by piece

You've been using the functionality of Zowe to issue commands and do all sorts of things through VS Code. In this challenge, we're just using 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, type the command **zowe console** and then hit enter. You can see there is an option to issue commands, as well as collect responses. Those are two more 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. The output goes on beyond what's captured in the screenshot above, and there are plenty of variations made available.

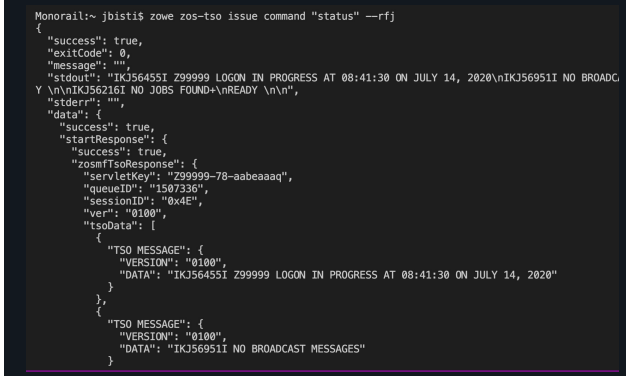
We're starting with the basics, don't worry, this will get a little more exciting in just a few more steps.

"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 VS Code plugin, a number of APIs, and the Zowe CLI which you're about to explore.

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's happening in these communities and help out whenever you see an opportunity.



7. Format for JSON

Make sure you have your jobs active 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!

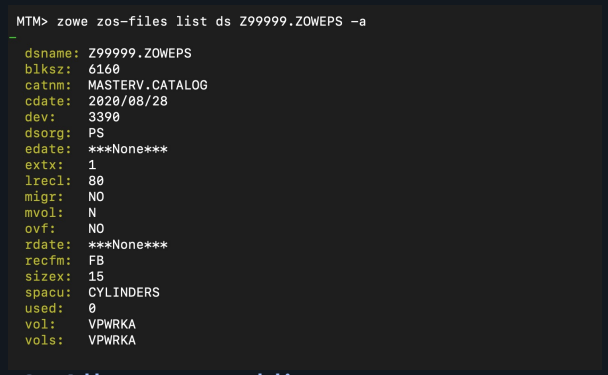
Now, issue the same command with **--rfj** (Response Format JSON) after it. Now you get the FULL output, and the output is in JSON format, which can be much more easily interpreted by programs that handle JSON format.

What about JSON

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

JSON stands for JavaScript Object Notation, 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 tends 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.

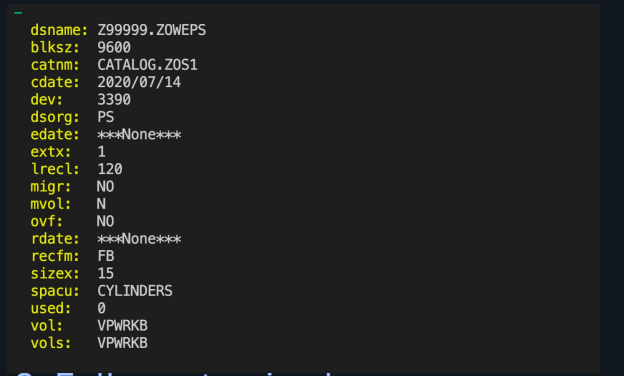


8. Allocate and list

Let's put this to use. Take a look at the *files* command group and use that to allocate (create) a sequential data set named **Zxxxxx.ZOWEPS** *(with your own userid, of course)*

Next, use another zowe cli command to show the attributes of the data set you just created. They should look similar to above.

If you get a timeout 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 data sets. The thing is, we made that dataset using a default set of values, and one of the great things about data sets is how customizable they are. Delete that data set (with another **files** command) and use the help (or online documentation at zowe.org) to **re-create that sequential data set** with some customized attributes.

First, we want the Record Length to be 120 instead of the default 80 (we've got some long records), and we want a block size of 9600.

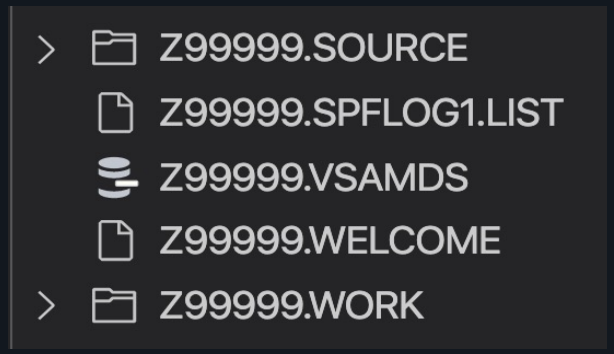
When you get it, you'll see a different readout for the Block Size and Record Length (LRECL), like in the screenshot above.

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

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

      $ 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
```

10. The keys to our data

One type of data set you have 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 dataset

You're 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 VS Code , you'll see a snazzy new icon. Curious yet? Let's proceed.

```
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
```

12. Load it up with records

Next, we are 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. Mockaroo.com has a nice data generator you can try out, though a few notes:

1) The first column (the "keys") must be in 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 dataset.

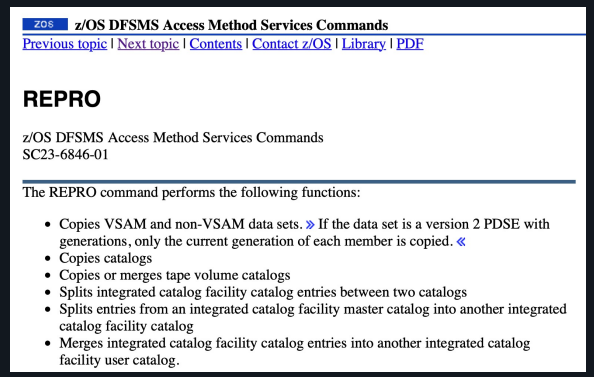
Download the sample **REPRO** JCL member from the ZXP.PUBLIC.JCL dataset to your personal workstation, placing it in the folder or directory you're currently working from. Note that if you created your own input dataset, you will need to edit the JCL to point to your source data set. Name your REPRO file as **repro.txt**

Use **zowe jobs submit local-file "repro.txt"** 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.

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 data sets, 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), ESDS (entry-sequenced), RRDS (relative record), and Linear (LDS). 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. That's enough for now, but if you're still hungry, here's some more to consider.



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:

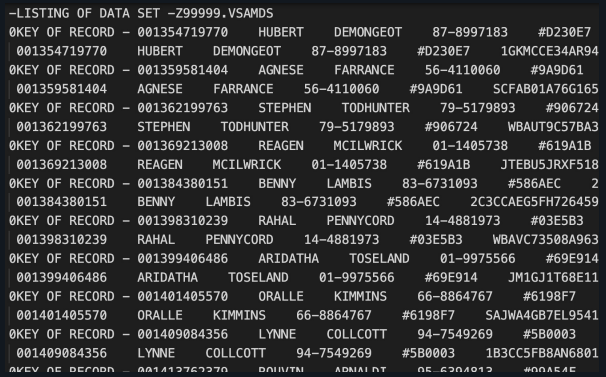
- Copies VSAM and non-VSAM data sets. » If the data set is a version 2 PDSE with generations, only the current generation of each member is copied. «
- Copies catalogs
- Copies or merges tape volume catalogs
- Splits integrated catalog facility catalog entries between two catalogs
- Splits entries from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

13. Let's take inventory

Let's talk about what we just did. The JCL runs IDCAMS, which is primarily used to manage VSAM data sets. Within IDCAMS, we're using the REPRO command to load a sequential data set into a VSAM-formatted data set. (There's a LOT of complexity happening here that we don't see, but just like before, 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 ones written in REXX)

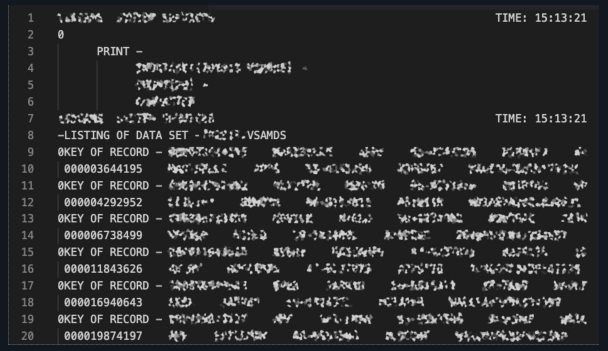
In reality, this data is not indexed very well, since each line is its own key, but if we dive into the particulars of building a VSAM cluster, you can see how the keys and record size can be specified.



```
-LISTING OF DATA SET -Z99999.VSAMDS
0KEY OF RECORD - 001354719770    HUBERT    DEMONGEOT    87-8997183    #D230E7
 001354719770    HUBERT    DEMONGEOT    87-8997183    #D230E7    1GKMCCE34AR94
0KEY OF RECORD - 001359581404    AGNESE    FARRANCE    56-4110060    #9A9D61
 001359581404    AGNESE    FARRANCE    56-4110060    #9A9D61    SCFAB01A76G165
0KEY OF RECORD - 001362199763    STEPHEN    TODHUNTER    79-5179893    #906724
 001362199763    STEPHEN    TODHUNTER    79-5179893    #906724    WBAUT9C57BA3
0KEY OF RECORD - 001369213008    REAGEN    MCILWRICK    01-1405738    #619A1B
 001369213008    REAGEN    MCILWRICK    01-1405738    #619A1B    JTEBU5JRXF518
0KEY OF RECORD - 001384380151    BENNY    LAMBIS    83-6731093    #586AEC    2
 001384380151    BENNY    LAMBIS    83-6731093    #586AEC    2C3CCAEG5FH726459
0KEY OF RECORD - 001398310239    RAHAL    PENNYCORD    14-4881973    #03E5B3
 001398310239    RAHAL    PENNYCORD    14-4881973    #03E5B3    WBAVC73508A963
0KEY OF RECORD - 001399406486    ARIDATHA    TOSELAND    01-9975566    #69E914
 001399406486    ARIDATHA    TOSELAND    01-9975566    #69E914    JM1GJ1T68E11
0KEY OF RECORD - 001401405570    ORALLE    KIMMINS    66-8864767    #6198F7
 001401405570    ORALLE    KIMMINS    66-8864767    #6198F7    SAJWA4GB7EL9541
0KEY OF RECORD - 001409084356    LYNNE    COLLCOTT    94-7549269    #5B0003
 001409084356    LYNNE    COLLCOTT    94-7549269    #5B0003    1B3CC5FB8AN6801
0KEY OF RECORD - 001413763270    ROUVIN    APNALDI    05-6394813    #99A54E
```

14. Printing out records

We're going to use one more IDCAMS command to look at our output, the aptly-named PRINT command and check out this example (hint) and pay attention to the CHARACTER parameter (hint hint) for your 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 data set in character format. Does that help? Don't be afraid to stop by the forums for some help.



```
                                                                    TIME: 15:13:21
0
     PRINT -
          [redacted]
          [redacted]
          [redacted]
                                                                    TIME: 15:13:21
-LISTING OF DATA SET - [redacted].VSAMDS
0KEY OF RECORD - [redacted]
 000003644195    [redacted]
0KEY OF RECORD - [redacted]
 000004292952    [redacted]
0KEY OF RECORD - [redacted]
 000006738499    [redacted]
0KEY OF RECORD - [redacted]
 000011843626    [redacted]
0KEY OF RECORD - [redacted]
 000016940643    [redacted]
0KEY OF RECORD - [redacted]
 000019874197    [redacted]
```

15. Make it count

Come for the Zowe CLI, stay for the VSAM and IDCAMS.
To complete this, we're looking for 3 things:
1) Your **Zxxxxx.ZOWEPS** sequential data set
2) Your **Zxxxxx.VSAMDS** VSAM data set
3) The first 20 lines of output from your IDCAMS PRINT command, copy/pasted into a sequential **Zxxxxx.OUTPUT.VSAMPRNT** data set. Don't write this data set directly from your JCL, use SYSPRINT, and copy/paste lines 1-20 of your SYSPRINT. We need the header. Refer to the screenshot above as a (lightly redacted) example.
When you have completed the task, submit **CHKAZCLI**

NICE JOB! LET'S RECAP

You came into this challenge *probably* not knowing much about ZCLI, 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" to "figure this out". Welcome to the big leagues, this is how we roll now.

NEXT UP...

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.