REXX1

REXX

Enterprise IT Duct Tape

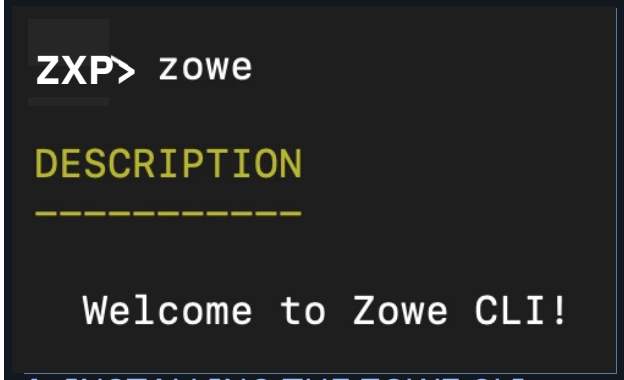
12 steps 60 minutes

THE CHALLENGE

In this challenge, you'll meet Rexx, a programming language known for its simplicity, power, and relative ease of use. We'll dive into how you run a Rexx program from a command line, as well as how to start up a TSO Address Space to run an interactive Rexx program. To interact with TSO, you will need the ZOWE Command Line Interface (CLI) installed.

BEFORE YOU BEGIN

This challenge requires some configuration to your terminal environment. Make sure you do the ZOWE CLI installation steps, then jump into the Rexx

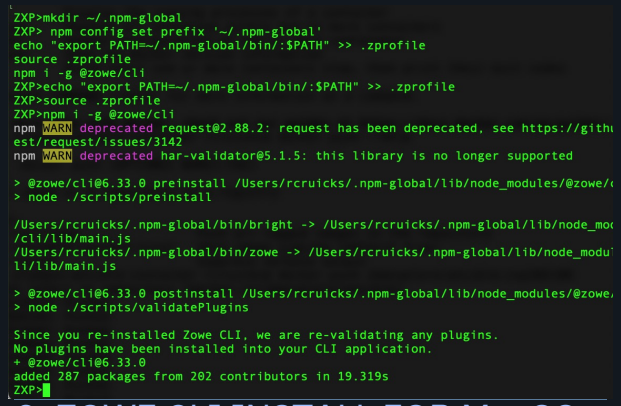


1. INSTALLING THE ZOWE CLI

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 Command Line Interface to work 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.

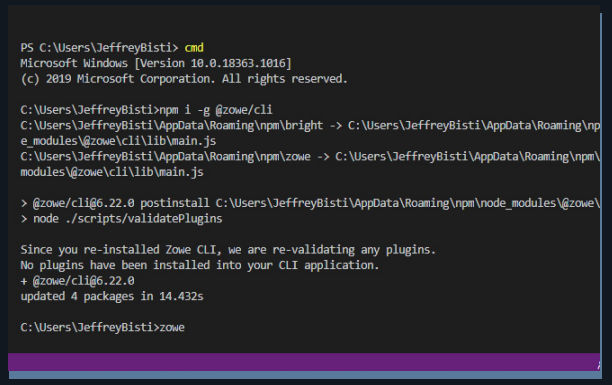


2. ZOWE CLI INSTALL FOR MacOS

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**

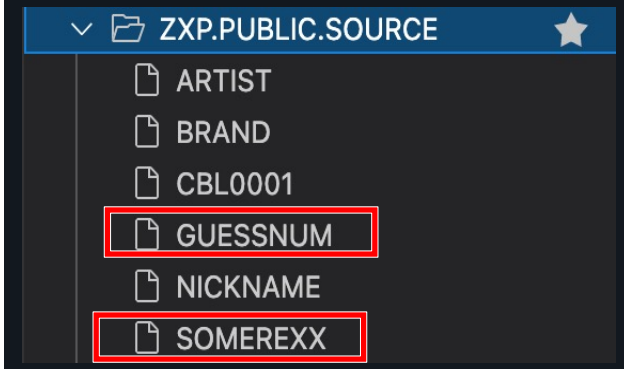
.



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.

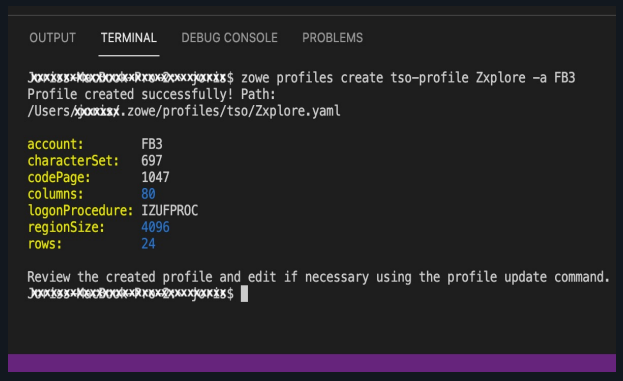
1: type **cmd** (this will change the shell to cmd from PowerShell)
1: **npm i -g @zowe/cli**
2: **zowe**



4. OPEN A TERMINAL

Start by copying the two files from ZXP.PUBLIC.SOURCE into your own SOURCE data set. Specifically, we're looking for SOMEREXX and GUESSNUM.

Next, open up a Terminal, just like you did for the USS commands, but don't SSH into anything. You should be able to type the command **zowe** from here and get some output from the program

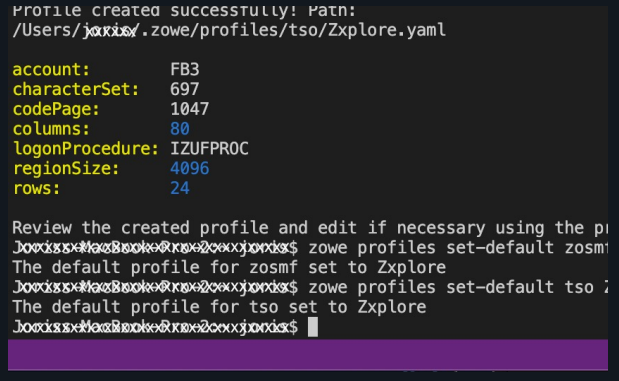


5. CREATE A TSO PROFILE

So far, we've been using a z/OSMF profile to connect. We need to create a TSO profile to issue TSO commands (more info on next page). Type this command:

zowe profiles create tso-profile zxplore -a FB3

Use whatever profile name you want, zxplore is just an example, and if you've been following our examples, it's nice to stay consistent.



6. SET DEFAULT PROFILES

At this point, it's also a good idea to make sure the right profiles are set as the defaults. They probably are, but just in case:
zowe profiles set-default zosmf zxplore
zowe profiles set-default tso zxplore

(Of course, use the profile names you chose)

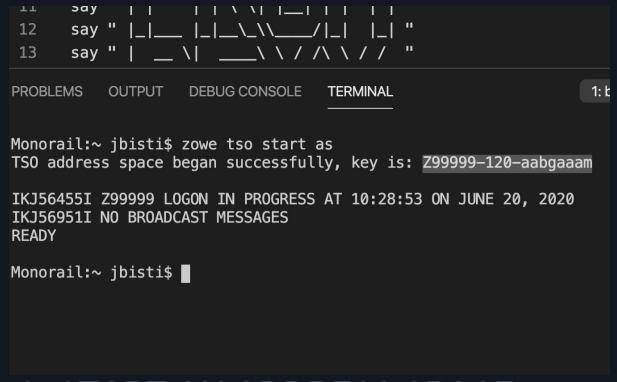
If you want to start over, use the **zowe profiles delete** command and follow the prompts. If you have other profiles from other mainframe activities, sometimes a restart of VS Code is required to make a profile switch take effect.

7. RUN SOME REXX

Type the following command:

zowe tso issue command "exec 'zXXXXX.source(somerexx)'" --ssm

Make sure to include all of the double and single quotes. Also, this may be the last time we point out that whenever you see Zxxxxx or Z99999, you need to input your own userid. We know most of you are tired of seeing that spelled out every time, but you wouldn't believe how many people get confused by not replacing sample text.

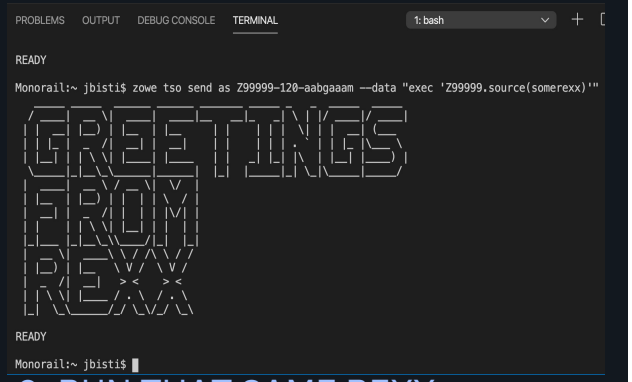


8. START AN ADDRESS SPACE

Start an address space with the following command:
zowe tso start as

This will create an address space for you, and tell you its key, which will start with your userid (as you can see above). You will use this key for the next few steps. Sometimes, after not being used for a while, a TSO session goes away. Just make another one using the same command.

You can stop an address space with the **stop** option



9. RUN THAT SAME REXX

Run the same Rexx program we did in step 7, but this time, direct the input towards the address space (and remember, you can probably press the Up arrow to recall previous commands)

zowe tso send as your-as-key --data "exec 'Zxxxxx.source(somerexx)'"

Note: (1) That's all on one line
(2) your key will be different
(3) this time we leave off the --ssm

You should get back the exact same response as before. The difference is that you're now issuing these commands through a semi-persistent TSO Address Space, which will make more sense in the next step.

"TSO? ADDRESS SPACE?"

TSO (Time Sharing Option) is another way that z/OS allows many many users to get access to data sets, run programs, and look at output. It is essentially the command-line interface for z/OS (when you're not using USS to access the UNIX side of things)

Think of an Address Space as a ticket that lets you start using system memory. An address space represents an enormous amount of memory, though the system will actually still control what lives in real on-chip memory, vs what gets moved (or *paged*) out to disk. Where your program goes its memory depends on how important it is, how it should be used, and if it will be shared with other programs. Address Spaces are a core part of z/OS, and you should read more about them when you get a minute:
https://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.zos.zconcepts/zconcepts_82.htm

```
READY

Monorail:~ jbisti$ zowe tso send as Z99999-120-aabgaaam --data "exec 'Z99999.source(guessnum)'"
I'm thinking of a number between 1 and 10.
What is your guess?
```

10. GUESS WHAT? ANOTHER EXEC

Do the same, but with the program **GUESSNUM**

This is a program that generates a random number between 1 and 10, and you have to guess that number. There may be other games you'd rather be playing right about now, but not all of them will teach you about Rexx and TSO, so keep that in mind when looking over at your PS4 controller.


```
Monorail:~ jbisti$ zowe tso send as Z99999-120-aabgaaam --data "exec 'Z99999.source(guessnum)'"
I'm thinking of a number between 1 and 10.
What is your guess?

Monorail:~ jbisti$ zowe tso send as Z99999-120-aabgaaam --data "1"
That's not it. Try again
What is your guess?

Monorail:~ jbisti$ zowe tso send as Z99999-120-aabgaaam --data "2"
That's not it. Try again
What is your guess?

Monorail:~ jbisti$ zowe tso send as Z99999-120-aabgaaam --data "3"
That's not it. Try again
What is your guess?

Monorail:~ jbisti$ zowe tso send as Z99999-120-aabgaaam --data "4"
You got it! And it only took you 4 tries!
READY
```

11. LIVE ACTION REXX CONVO

Send your guesses to the program by replacing everything between the double-quotes with a number. You can see here that we're using the persistent address space key to ensure our input goes to the TSO address space, which is sitting there waiting for our input.

If you get it right on the first try, congratulations! Feel free to start the program again and make sure you can see it go through the "Try again" steps.

Now submit your completion check – CHKREXX1

```
1  /* REXX */
2  say "I'm thinking of a number between 1 and 10."
3  secret = RANDOM(1,10)
4  tries = 1
5
6  do while (guess \= secret)
7      say "What is your guess?"
8      pull guess
9      if (guess \= secret) then
10     do
11         say "That's not it. Try again"
12         tries = tries + 1
13     end
14 end
15 say "You got it! And it only took you" tries "tries!"
16 exit
```

12. SEE HOW CUTE THE CODE IS

If you haven't already, load up the code for that program in your VS Code editor. It's not a complicated program by any means, but you might be noticing just how simple this Rexx code really is. 16 lines of code, including a comment and a blank, and yes, those "say" and "pull" commands really do what you think they do.

You can see why people love this language. It's also got what I consider to be the world's greatest logo for a programming language. Look at it. Just. Look. At. It.

NICE JOB! LET'S RECAP

Now you're getting into the swing of things. You're interacting with a Rexx program, running in a TSO address space, and you're doing that through the zowe command.

You've probably also learned quite a bit about the Rexx language, and may have even done some extra reading about Address Spaces. Everything in here will help you become a skilled mainframe professional.

NEXT UP...

The iron is most definitely smoldering, and you're probably becoming a fan of Rexx. In the advanced channel, you'll find REXX2 where you'll write your own code from scratch and implement some file reads/writes. For now, though, let's continue to the next challenge in Fundamentals.