include::attributes.txt[]

// Attributes

[.topic] [#configure-access-sso-example-cli] = Example: Authenticate with IAM Identity Center automatic token refresh for use with the {aws} CDK CLI :info\_titleabbrev: Example: Authenticate with IAM Identity Center automatic token refresh :keywords: {aws}, {aws} CDK, {aws} CDK CLI, Programmatic access, Security credentials, IAM, IAM Identity Center

== [abstract]

## In this example, we configure the {aws} Command Line Interface to authenticate our user with the {aws} IAM Identity Center token provider configuration. The SSO token provider configuration lets the {aws} CLI automatically retrieve refreshed authentication tokens to generate short-term credentials that we can use with the {aws} Cloud Development Kit ({aws} CDK) Command Line Interface ({aws} CDK CLI).

// Content start

In this example, we configure the {aws} Command Line Interface ({aws} CLI) to authenticate our user with the {aws} IAM Identity Center token provider configuration. The SSO token provider configuration lets the {aws} CLI automatically retrieve refreshed authentication tokens to generate short-term credentials that we can use with the {aws} Cloud Development Kit ({aws} CDK) Command Line Interface ({aws} CDK CLI).

[#configure-access-sso-example-cli-prerequisites] == Prerequisites

This example assumes that the following prerequisites have been completed:

* Prerequisites required to get set up with {aws} and install our starting CLI tools. For more information, see xref:configure-access-prerequisites[Prerequisites].
* IAM Identity Center has been set up by our organization as the method of managing users.
* At least one user has been created in IAM Identity Center.

[#configure-access-sso-example-cli-configure] == Step 1: Configure the {aws} CLI

For detailed instructions on this step, see https://docs.aws.amazon.com/cli/latest/userguide/sso-configure-profile-token.html[Configure the {aws} CLI to use IAM Identity Center token provider credentials with automatic authentication refresh] in the *{aws} Command Line Interface User Guide*.

We sign in to the {aws} access portal provided by our organization to gather our IAM Identity Center information. This includes the *SSO start URL* and *SSO Region*.

Next, we use the {aws} CLI aws configure sso command to configure an IAM Identity Center profile and sso-session on our local machine:

== [source,bash,subs=“verbatim,attributes”]

$ aws configure sso SSO session name (Recommended): ++++++SSO start URL [None]: <https://my-sso-portal.awsapps.com/start> SSO region [None]: ++++++SSO registration scopes [sso:account:access]: ++++++—-++++++++++++++++++

The {aws} CLI attempts to open our default browser to begin the login process for our IAM Identity Center account. If the {aws} CLI is unable to open our browser, instructions are provided to manually start the login process. This process associates the IAM Identity Center session with our current {aws} CLI session.

After establishing our session, the {aws} CLI displays the {aws} accounts available to us:

== [source,none,subs=“verbatim,attributes”]

There are 2 {aws} accounts available to you.

DeveloperAccount, developer-account-admin@example.com (<123456789011>) ProductionAccount, production-account-admin@example.com (<123456789022>) — \_\_\_\_

We use the arrow keys to select our *DeveloperAccount*.

Next, the {aws} CLI displays the IAM roles available to us from our selected account:

== [source,none,subs=“verbatim,attributes”]

Using the account ID <123456789011> There are 2 roles available to you.

ReadOnly FullAccess — \_\_\_\_

We use the arrow keys to select *FullAccess*.

Next, the {aws} CLI prompts us to complete configuration by specifying a default output format, default {aws} Region, and name for our profile:

== [source,none,subs=“verbatim,attributes”]

CLI default client Region [None]: ++++++++++++CLI default output format [None]: ++++++++++++CLI profile name [123456789011\_FullAccess]: ++++++++++++—-++++++++++++++++++++++++++++++++++++

The {aws} CLI displays a final message, showing how to use the named profile with the {aws} CLI:

== [source,none,subs=“verbatim,attributes”]

To use this profile, specify the profile name using –profile, as shown:

== aws s3 ls –profile ++++++++++++

After completing this step, our config file will look like the following:

== [source,none,subs=“verbatim,attributes”]

[profile ++++++] sso\_session = ++++++sso\_account\_id = <123456789011> sso\_role\_name = ++++++region = ++++++output = ++++++++++++++++++++++++++++++++++++

[sso-session ++++++] sso\_region = ++++++sso\_start\_url = <https://my-sso-portal.awsapps.com/start> sso\_registration\_scopes = —-++++++++++++

We can now use this sso-session and named profile to request security credentials.

[#configure-access-sso-example-cli-credentials] == Step 2: Use the {aws} CLI to generate security credentials

For detailed instructions on this step, see https://docs.aws.amazon.com/cli/latest/userguide/sso-using-profile.html[Use an IAM Identity Center named profile] in the *{aws} Command Line Interface User Guide*.

We use the {aws} CLI aws sso login command to request security credentials for our profile:

== [source,bash,subs=“verbatim,attributes”]

$ aws sso login –profile ++++++—-++++++

The {aws} CLI attempts to open our default browser and verifies our IAM log in. If we are not currently signed into IAM Identity Center, we will be prompted to complete the sign in process. If the {aws} CLI is unable to open our browser, instructions are provided to manually start the authorization process.

After successfully logging in, the {aws} CLI caches our IAM Identity Center session credentials. These credentials include an expiration timestamp. When they expire, the {aws} CLI will request that we sign in to IAM Identity Center again.

Using valid IAM Identity Center credentials, the {aws} CLI securely retrieves {aws} credentials for the IAM role specified in our profile. From here, we can use the {aws} CDK CLI with our credentials.

[#configure-access-sso-example-cli-cdk] == Step 3: Use the CDK CLI

With any CDK CLI command, we use the xref:ref-cli-cmd-options-profile[--profile] option to specify the named profile that we generated credentials for. If our credentials are valid, the CDK CLI will successfully perform the command. The following is an example:

== [source,none,subs=“verbatim,attributes”]

$ cdk diff –profile ++++++Stack CdkAppStack Hold on while we create a read-only change set to get a diff with accurate replacement information (use –no-change-set to use a less accurate but faster template-only diff) Resources [-] {aws}::S3::Bucket amzn-s3-demo-bucket amzn-s3-demo-bucket5AF9C99B destroy++++++

Outputs [-] Output BucketRegion: {“Value”:{“Ref”:“{aws}::Region”}}

== ✨ Number of stacks with differences: 1

When our credentials expire, an error message like the following will display:

== [source,none,subs=“verbatim,attributes”]

$ cdk diff –profile ++++++Stack CdkAppStack++++++

== Unable to resolve {aws} account to use. It must be either configured when you define your CDK Stack, or through the environment

To refresh our credentials, we use the {aws} CLI aws sso login command:

== [source,bash,subs=“verbatim,attributes”]

$ aws sso login –profile ++++++—-++++++