Configure Secret Management Support

Last Modified on 07/26/2022 9:05 am EDT

V10.0-V10.1

Prerequisites

- Basic understanding of Cora Orchestration configuration.
- Basic understanding of the external secret stores, AWS Secrets Manager, or Azure Key Vault, whichever you want to use.
- Knowledge of creating secret keys in the secret stores.

Overview

Cora Orchestration supports external secret stores to store and fetch values like user credentials, connection strings, database credentials, API keys, OAuth tokens, and other secrets for the configuration files at runtime. With this functionality you need not hardcode the sensitive information in plain text or encrypted text. With secret stores you can also control the permission-based access to the sensitive information.

Following are the three secret stores supported:

- AWS Secrets Manager
- Azure Key Vault
- System Environment Variables

Starting from V10.1, to choose a specific secret store for Cora Orchestration, and connect to the secret store, you need to run the Add-CoraOrchestrationSecretEnvironmentVariables PowerShell function.

For V10.0, configuring the secret store is done manually.
To choose the specific secret store, you need to add the `sequence:secrets:providerTypes` environment variable in your system environment variables, and to connect to the secret store, you need to configure the store specific environment variables in the system.

All the store specific environment variables are listed in the sections below.
For the list of secret keys required for Cora Orchestration, see the Secret keys section below.

AWS Secrets Manager environment variables

Environment variable	Description
sequence:secrets:providerTypes	The secret store type Value: **AWSSecretManager**
sequence:secrets:awsAccessKey	The access key
sequence:secrets:awsSecretKey	The secret key
sequence:secrets:awsRegion	The region for which secret store is being set

Environment variable	Description
sequence:secrets:awsUseSecretNameAsKeyPrefix (Optional)	When True, will generate keys with secret name as prefix: "secretName:secretKey". When False, will generate keys without secret name as prefix: "secretKey".
sequence:secrets:awsKeyPrefixFilter (Optional)	The prefix that all keys must include.
sequence:secrets:awsAcceptedSecretArns (Optional)	The list of identifiers for the secrets that are to be retrieved. The secret ARN (full or partial) and secret name are supported. For example: MySecretFullARN-abcxyz;MySecretPartialARN;MySecretUniqueName
sequence:secrets:awsPollingInterval (Optional)	The waiting time before refreshing the secrets. If null, secrets will not be refreshed. For example, 00:15:00 for 15 minutes.
sequence:secrets:awsSecretNamesFilter (Optional)	The list of secret names that get passed to the client to filter the listed secrets before returning them. For example, secret1;secret2

Azure Key Vault environment variables

Environment variable	Description
sequence:secrets:providerTypes	The secret store type Value: **AzureKeyVault**
sequence:secrets:azureKeyVaultUri	The Azure Uniform Resource Identifier of the key vault
sequence:secrets:azureKeyVaultTenantId	The ID of the tenant (directory) where the AD application is registered
sequence:secrets:azureKeyVaultClientId	The ID of the application (client) that you created to read the secrets
sequence:secrets:azureKeyVaultClientSecret	The secret for the Azure Active Directory application
sequence:secrets:azureKeyVaultSecretKeyPrefix (Optional)	The prefix for the names of the secrets in the vault

System environment variables

If you don't want to use an external secret store, you can use your system environment variables to store secrets.

Environment variable	Value/Description
sequence:secrets:providerTypes	The secret store type Value: **EnvironmentVariables**

Secret keys

The following are the secret keys and their values you need to store in your secret store.

Secret key	Description	Value
sequence:persistence:database:provider	Database provider name	Microsoft.Data.SqlClient
sequence:persistence:database:credentials	Database credentials	user id=sa;password=sa;
sequence:persistence:database:connectionString	Database connection string	For example, MultipleActiveResultSets=true;initial catalog=DBName;persist security info=True;data source=DBserverName;packet size=4096;
sequence:messageBus:connections:defaultConnectionName	Message bus connection name	• SqlServiceBroker • ActiveMQ
sequence:messageBus:connections:activeMQ:credentials	ActiveMQ credentials, if you have added ActiveMQ as default connection name	user id=mb1;password=sd;
sequence:messageBus:connections:activeMQ:connectionString	ActiveMQ connection string, if you have added ActiveMQ as default connection name	For example, Server=failover:(tcp://192.168.xx.x:00000);Username=usr1;Password=pswd1;
sequence:cryptography:sha256:salt	The sha256 salt to prevent identical passwords **NOTE** When you upgrade, this value should not be changed.	Base64string
sequence:cryptography:rijndael:key	The Rijndael key **NOTE** When you upgrade, this value should not change.	Base64string

Secret key	Description	Value
sequence:cryptography:rijndael:salt	The Rijndael salt to prevent identical passwords **NOTE** When you upgrade, this value should not change.	Base64string