Azure Key Vault

# Packages

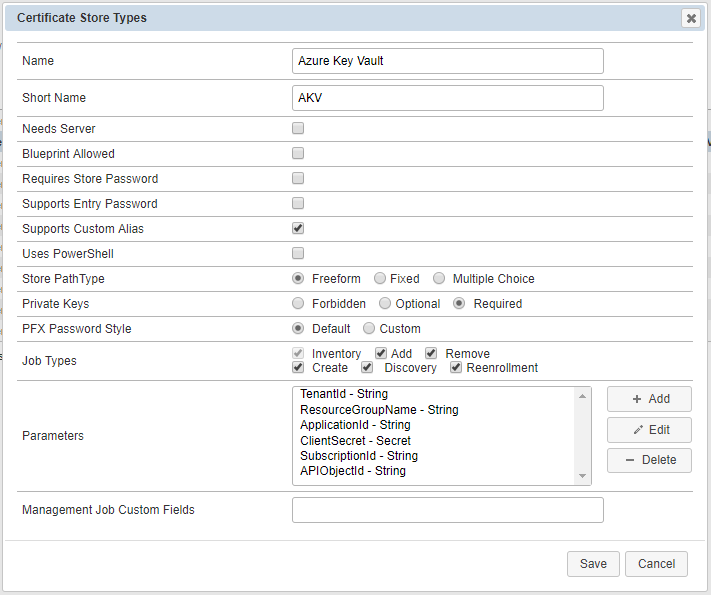
* GAC
  + System.Net.Http
  + System.Security.Cryptography.X509Certificates
  + System.Threading.Tasks
* NuGet
  + Microsoft.Azure.KeyVault v3.0.0 (<https://github.com/Azure/AutoRest>)
  + Newtonsoft.Json.Linq v6.0.0 (Dependency from above)
  + Microsoft.IdentityModel.Clients.ActiveDirectory v3.19.8 (<https://aka.ms/adalnet>)
  + Microsoft.AspNet.WebApi.Client v5.2.3 (<https://www.asp.net/web-api>)
  + Microsoft.Azure.Management.KeyVault v2.4.1 (<https://github.com/Azure/azure-sdk-for-net>)
* CSS
  + CSS.CMS.Extensions.Agents;
  + CSS.PlatformServices;

# Azure Configuration

1. Create a new Azure Key Vault
   1. Select (All networks can access, unless you plan to configure routing rules and firewalls – not covered)
2. Collect the base Key Vault URL (available as the DNS Name under the details of the Key Vault resource)
3. Register a new API Application in AAD
   1. Use a URL like <https://localhost:44313>, you won’t need this
4. Add a key to the API Application created in (2)
   1. Save this key, it disappears after navigating away from the save page
5. Collect the Tenant ID (Also called Directory ID available under AAD -> Properties)
6. Collect the API Application ID (Details of the API Application Registration in AAD)
7. Collect your Subscription ID (Details of your Subscription)
8. Add your API as a principal to your Key Vault resource
   1. Open the details of your Key Vault resource
   2. Click on Access Policies
   3. Click add new
   4. Ensure it has the proper permissions
   5. Select the application registered in (3) as the principal
   6. Click Ok
   7. Click Save
9. Add your API principal to your subscription
   1. Go to the subscription details
   2. Click on “Access Control”
   3. Click add
   4. Select “Owner”
   5. Select the principal of the application registered in (3)
   6. Click save

# CMS Certificate Store Configuration

## Minimum Required Certificate Store Definition Parameters



\*StorePath :: The URL of the vault you are operating on (If creating a new vault, this should be https://<Your\_Vault\_Name>.vault.azure.net/)

# Parameters

* \*TenantId :: The ID of the Azure Active Directory (obtained from Properties menu of the subscription’s Azure Active Directory – on this page it is called “Directory ID”)
* \*ResourceGroupName :: The resource group name to which the Key Vault(s) belong
* \*VaultName :: Name of the specific vault to add/delete certificates from. (Remark, if creating a new store, this will be used for its name)
* \*ApplicationId :: The ID of the API Application you are using to communicate with the Azure Key Vault (obtained from the App registration found under the App registrations menu of the subscription’s Azure Active Directory)
* \*ClientSecret :: The secret value given to you when you created a key for the API application in step *Azure Configuration (4)*
* \*SubscriptionId :: The ID of the subscription the Azure Key Vault(s) reside in (obtained from the Properties menu of the Subscription under the Subscriptions menu)
* APIObjectId :: The object ID of the API Application (obtained from the App registration found under the App registrations menu of the subscription’s Azure Active Directory) (required if a vault name is supplied)

\* Denotes a required value

# Dependent Assemblies

Add the following to …/Certificate Management System Agent/CMSAgent.exe.config

* <dependentAssembly>

<assemblyIdentity name="Microsoft.Rest.ClientRuntime" publicKeyToken="31bf3856ad364e35" culture="neutral" />

<bindingRedirect oldVersion="0.0.0.0-2.0.0.0" newVersion="2.0.0.0" />  
</dependentAssembly>

# Talking Points

* While Azure Key Vault will automatically renew certs from DigiCert, GlobalSign, and D-Trust, they only send emails for certificates from other certificate providers
* You have the option to specify that the key pair be generated as non-exportable and generated inside an HSM. If you do this then you cannot distribute your certificate and private key to your app nodes as illustrated in the end-to-end flow example above.
* The Azure App Services platform periodically polls your key vault to check if there is an updated certificate. If it finds one it reads the new one and rebinds SSL/TLS for your app.