# svs-oauth2clientcredentials-aad Middleware Component Info Sources

by George Stevens, Solid Value Software

3/2/21. Revision 1 on 3/3/21

One useful by product of my work learning to use the Dapr Pub/Sub component with the Confidential Client Credential middleware component is that I am also making a sample of this for use by the public. Once I get the sample working I will finalize this repo and consult with the Dapr samples maintainer about the details of contributing a sample. Thus, the main concern of this document is to provide an accurate mapping of the information required to interact with an Azure Service Bus topic via the Client Credentials Flow mapped onto the Dapr middleware.http.oauth2clientcredentials component. In other words, exactly what information goes into each of the “blank fields” of the below component.

## Information Required for the Native Code Implementation of the Client Credentials Flow

I have completed the implementation of a native .NET C# version of the Client Credentials flow for Azure Active Directory (aad) publishing items to an Azure Service Bus Topic. The primary Microsoft example code sources I followed are below, and will be useful to review while reading this document since they define many of the Azure and .NET specific terms and techniques:

1. <https://github.com/Azure/azure-service-bus/blob/master/samples/DotNet/Microsoft.Azure.ServiceBus/RoleBasedAccessControl/Program.cs>
2. <https://docs.microsoft.com/en-us/azure/active-directory/develop/scenario-daemon-app-configuration?tabs=dotnet>
3. <https://docs.microsoft.com/en-us/azure/active-directory/develop/scenario-daemon-acquire-token?tabs=dotnet#acquiretokenforclient-api> and its related documents.

Below is a list of the information required to interact (currently publish only) with an Azure Service Bus (ASB) topic , plus where I currently believe the information should be placed in the middleware component yaml definition. Note that much of the information was obtained via the App Registrations process in the Azure Portal for AAD, but some info was obtained from the above code examples and documents.

In the below list I use the name of each item as it is generally used in the Azure documentation. Sometimes that is the same as in the below svs-oauth2clientcredentials-aad middleware component, and other time not so much.

**ClientId** or AppId: Obtained from the Azure Portal AAD App Registrations, Overview screen.

The ClientId value (a Guid) goes into the value field of -name:clientId below. ***OK***.

**ClientSecret**: Obtained from the Azure Portal AAD App Registrations, Certificates and Secrets screen.

The ClientSecret value (a Guid) goes into the value field of -name:clientSecret below. ***OK***.

This document is highly useful for the following items -- <https://docs.microsoft.com/en-us/azure/active-directory/develop/scenario-daemon-app-configuration?tabs=dotnet>

**Scopes** or claims (a comma separates each individual scope string): Obtained from code examples and documents.

The Service Principal for my app has Azure Service Bus Data Sender permissions registered in the ASB Topic’s Access Control (IAM) under the Role Assignments screen. This is used for fine grained Role Based Access Control performed by Azure based on Role Assignments rather than on fine grained claims in an Access Token.

The “claims” I am using in my C# code is a single claim (scope), <https://servicebus.azure.net/.default>. The ASB itself will verify that the Service Principal sending the request has been previously authorized to do the specific Send, Receive, etc. ASB Topic operation. This scope string goes into the value field of -name:scopes below. ***OK***.

**TenantId** or Directory(aad)Id: Obtained from the Azure Portal AAD App Registrations, Authentication screen.

WHERE DOES THIS GO in the component yaml? My code sample is for a line-of-business app which uses a single Azure Active Directory, aka tenant. A single tenant is represented by a Guid.

It seems that tenantId:tenantIdValue name/value pair should be in the component under endpointParamsQuery. But MAYBE NOT since **tenantId is concatenated with another string to produce the “authority” value**, shown directly below. ***NOT OK***. ***DO I SET TenantId***? **Or instead use in in the below authority**?

**Authority**: Obtained from code examples.

WHERE DOES THIS GO in the component yaml? My code sample uses this to build an instance of IConfidentialClientApplication using the below code:

IConfidentialClientApplication app =

ConfidentialClientApplicationBuilder

.Create(m\_PubCredsNInfo.ClientId)

.WithAuthority(authority)

.WithClientSecret(m\_PubCredsNInfo.ClientSecret)

.Build();

The value used for “authority” in my code is the string <https://login.windows.net/> **concatenated** with TenantId, as shown by this line of code from my sample:

string authority = $"{AadAuthnAuthzConstants.LoginForAuthNAuthority}{m\_PubCredsNInfo.TenantId}";

or more clearly

authority = **“**[**https://login.windows.net/**](https://login.windows.net/)**TenantIdGuidValue”**;

Thus, It seems that authority:authorityValue name/value pair should be in the component under endpointParamsQuery. ***NOT OK***. ***Please PROVIDE AN EXAMPLE*** of the value to set this, and the format.

**Connection String** or **Service Bus Namespace Domain Name**: Obtained from my code examples: **Revision 1 on 3/3/21**.

However*, this information may only be needed in the PubSub component*, and not in the Client Credential Middleware. *But just in case*, here it is: My code uses environment variables that contain either the

Service Bus Name Space domain name = svsdaprasbns.servicebus.windows.net, for example.

Or

the connection string = Endpoint=sb://svsdaprasbns.servicebus.windows.net/, for example.

Thus, it seems one of these name/value pairs should be in the component under endpointParamsQuery. ***NOT OK***. ***Please PROVIDE AN EXAMPLE*** of the value to set this, and the format

And that’s it. I furnish no other information in my code samples and neither do the code examples I followed.

But what about the following items? They are from the below svs-oauth2clientcredentials-aad middleware component that were NOT required by my code. What should I do about these?

**authStyle** -- Clearly the authStyle should be 1. ***OK***.

**Token Url** -- Note that in the examples this is **required only in interactive clients**, and not confidential clients. For example in the IConfidentialClientApplication builder for an interactive app it does the following:

.WithRedirectUri(ConfigurationManager.AppSettings["redirectURI"])

But that is not used in confidential clients as you can see from the above IConfidentialClientApplication builder code. ***NOT OK***. ***TO WHAT VALUE SHOULD I SET THIS***?

**HeaderName** *--* ***NOT OK***. ***TO WHAT VALUE SHOULD I SET THIS***?

## Information Required for the Dapr svs-oauth2clientcredentials-aad Middleware Component Implementation of Client Credentials Flow

Here is the Dapr OAuth (Client Credentials) svs-oauth2clientcredentials-aad middleware component patterned after that shown in the Dapr Documentation at [https://docs.dapr.io/operations/security/oauth/#define-a-client-credentials-grant-component](https://docs.dapr.io/operations/security/oauth/%23define-a-client-credentials-grant-component). Below, only the metadata:name: svs-oauth2clientcredentials-aad has been changed from the original at the above link.

apiVersion: dapr.io/v1alpha1

kind: Component

metadata:

name: svs-oauth2clientcredentials-aad

spec:

type: middleware.http.oauth2clientcredentials

version: v1

metadata:

- name**: clientId**

value: "<your client ID>"

- name: **clientSecret**

value: "<your client secret>"

- name: **scopes**

value: "<comma-separated scope names>"

- name: **tokenURL**

value: "<token issuing URL>"

- name: **headerName**

value: "<header name under which the secret token is saved>"

- name: **endpointParamsQuery**

value: "<list of additional key=value settings separated by ampersands or semicolons forwarded to the token issuing service>"

# authStyle:

# "0" means to auto-detect which authentication

# style the provider wants by trying both ways and caching

# the successful way for the future.

# "1" sends the "client\_id" and "client\_secret"

# in the POST body as application/x-www-form-urlencoded parameters.

# "2" sends the client\_id and client\_password

# using HTTP Basic Authorization. This is an optional style

# described in the OAuth2 RFC 6749 section 2.3.1.

- name: **authStyle**

value: "<see comment>"