1. Authorizing access to data in Azure (for SBS and SA only):
   1. Shared access signature
      1. What is?

Overview of SAS

Shared Access Signatures are a claims-based authorization mechanism using simple tokens. Using SAS, keys are never passed on the wire. Keys are used to cryptographically sign information that can later be verified by the service. SAS can be used similar to a username and password scheme where the client is in immediate possession of an authorization rule name and a matching key. SAS can also be used similar to a federated security model, where the client receives a time-limited and signed access token from a security token service without ever coming into possession of the signing key.

1. How it Works
   1. Go to the portal and open the resource that you want to geneate the SAS token.
   2. Select the Shared access signature in Settings section:
   3. Specify the allowed service to be access;
   4. The resource type;
   5. The permission;
   6. The Start and expiry date/time.
   7. Optionally you can restrict the acess by IP and Protocol.
   8. Click in "Generate SAS and connection string" button

Graphical user interface, text, application, email

Description automatically generated

1. Copy the SAS token. Each service has its own SAS link.

Background pattern

Description automatically generated

1. Anonymous public read access
   1. What is?

Azure Storage supports optional anonymous public read access for containers and blobs. By default, anonymous access to your data is never permitted. Unless you explicitly enable anonymous access, all requests to a container and its blobs must be authorized. When you configure a container's public access level setting to permit anonymous access, clients can read data in that container without authorizing the request.

1. How it Works

To allow or disallow public access for a storage account in the Azure portal, follow these steps:

1. Navigate to your storage account in the Azure portal.
2. Locate the Configuration setting under Settings.
3. Set Blob public access to Enabled or Disabled.

Graphical user interface, text, application

Description automatically generated

1. Shared Key authorization (storage account key)
   1. What is?

Every request made against a storage service must be authorized, unless the request is for a blob or container resource that has been made available for public or signed access. One option for authorizing a request is by using Shared Key. To authenticate you will only provide the account name and the account key.

1. How it Works.

To get the key, follow these steps:

1. Navigate to your storage account in the Azure portal.
2. Locate Access keys under Settings
3. Copy the key from as below. (You can use 2 different keys. Also you can re-generate the keys clicking in the refresh button)

A picture containing indoor, sitting, book, television

Description automatically generated

1. On-premises Active Directory Domain Services (AD DS, or on-premises AD DS) authentication (preview)
   1. What is?

Azure Files supports identity-based authentication over Server Message Block (SMB) through on-premises Active Directory Domain Services (AD DS) and Azure Active Directory Domain Services (Azure AD DS). This article focuses on how Azure file shares can use domain services, either on-premises or in Azure, to support identity-based access to Azure file shares over SMB. Enabling identity-based access for your Azure file shares allows you to replace existing file servers with Azure file shares without replacing your existing directory service, maintaining seamless user access to shares.

Azure Files enforces authorization on user access to both the share and the directory/file levels. Share-level permission assignment can be performed on Azure Active Directory (Azure AD) users or groups managed through the Azure role-based access control (Azure RBAC) model. With RBAC, the credentials you use for file access should be available or synced to Azure AD. You can assign Azure built-in roles like Storage File Data SMB Share Reader to users or groups in Azure AD to grant read access to an Azure file share.

At the directory/file level, Azure Files supports preserving, inheriting, and enforcing Windows DACLs just like any Windows file servers. You can choose to keep Windows DACLs when copying data over SMB between your existing file share and your Azure file shares. Whether you plan to enforce authorization or not, you can use Azure file shares to back up ACLs along with your data.

1. How it Works

A picture containing chart

Description automatically generated

1. Azure Active Directory (Azure AD) integration: Authenticate from an application
   1. What is?

When a security principal (a user, group, or application) attempts to access a Service Bus entity, the request must be authorized. With Azure AD, access to a resource is a two-step process.

1. First, the security principal’s identity is authenticated, and an OAuth 2.0 token is returned. The resource name to request a token is <https://servicebus.azure.net>.
2. Next, the token is passed as part of a request to the Service Bus service to authorize access to the specified resource.

The authentication step requires that an application request contains an OAuth 2.0 access token at runtime. If an application is running within an Azure entity such as an Azure VM, a virtual machine scale set, or an Azure Function app, it can use a managed identity to access the resources. To learn how to authenticate requests made by a managed identity to Service Bus service, see Authenticate access to Azure Service Bus resources with Azure Active Directory and managed identities for Azure Resources.

The authorization step requires that one or more Azure roles be assigned to the security principal. Azure Service Bus provides Azure roles that encompass sets of permissions for Service Bus resources. The roles that are assigned to a security principal determine the permissions that the principal will have. To learn more about assigning Azure roles to Azure Service Bus, see Azure built-in roles for Azure Service Bus.

Native applications and web applications that make requests to Service Bus can also authorize with Azure AD. This article shows you how to request an access token and use it to authorize requests for Service Bus resources.

1. How it Works
   1. In the Azure portal, navigate to your Service Bus namespace. Select Access Control (IAM) on the left menu to display access control settings for the namespace. If you need to create a Service Bus namespace, follow instructions from this article: Create a Service Bus Messaging namespace.

Graphical user interface, text, application, email

Description automatically generated

1. Select the Role assignments tab to see the list of role assignments. Select the Add button on the toolbar and then select Add role assignment.

Graphical user interface, text

Description automatically generated

1. On the Add role assignment page, do the following steps:
   1. Select the Service Bus role that you want to assign.
   2. Search to locate the security principal (user, group, service principal) to which you want to assign the role.
   3. Select Save to save the role assignment.

Graphical user interface, application

Description automatically generated

1. The identity to whom you assigned the role appears listed under that role. For example, the following image shows that Azure-users is in the Azure Service Bus Data Owner role.

Graphical user interface, text, application, email

Description automatically generated

1. Azure Active Directory Domain Services (Azure AD DS) authentication
   1. What is?

Azure Active Directory Domain Services (AD DS) provides managed domain services such as domain join, group policy, lightweight directory access protocol (LDAP), and Kerberos / NTLM authentication. You use these domain services without the need to deploy, manage, and patch domain controllers (DCs) in the cloud.

An Azure AD DS managed domain lets you run legacy applications in the cloud that can't use modern authentication methods, or where you don't want directory lookups to always go back to an on-premises AD DS environment. You can lift and shift those legacy applications from your on-premises environment into a managed domain, without needing to manage the AD DS environment in the cloud.

Azure AD DS integrates with your existing Azure AD tenant. This integration lets users sign in to service and applications connected to the managed domain using their existing credentials. You can also use existing groups and user accounts to secure access to resources. These features provide a smoother lift-and-shift of on-premises resources to Azure.

1. How it Works

Permission is very similar as the previous method because you can integrate ADDS with you Azure AD and assign permissions using Role Assignment

1. Azure Active Directory (Azure AD) integration: Authenticate with managed identities
   1. What is?

Managed identities for Azure resources is a cross-Azure feature that enables you to create a secure identity associated with the deployment under which your application code runs. You can then associate that identity with access-control roles that grant custom permissions for accessing specific Azure resources that your application needs.

With managed identities, the Azure platform manages this runtime identity. You do not need to store and protect access keys in your application code or configuration, either for the identity itself, or for the resources you need to access. A Service Bus client app running inside an Azure App Service application or in a virtual machine with enabled managed entities for Azure resources support does not need to handle SAS rules and keys, or any other access tokens. The client app only needs the endpoint address of the Service Bus Messaging namespace. When the app connects, Service Bus binds the managed entity's context to the client in an operation that is shown in an example later in this article. Once it is associated with a managed identity, your Service Bus client can do all authorized operations. Authorization is granted by associating a managed entity with Service Bus roles.

1. How it Works

To use Service Bus with managed identities, you need to assign the identity the role and the appropriate scope. The procedure in this section uses a simple application that runs under a managed identity and accesses Service Bus resources.

Here we're using a sample web application hosted in Azure App Service. For step-by-step instructions for creating a web application, see Create an ASP.NET Core web app in Azure

Once the application is created, follow these steps:

1. Go to Settings and select Identity.
2. Select the Status to be On.
3. Select Save to save the setting.

Graphical user interface, text, application

Description automatically generated

Once you've enabled this setting, a new service identity is created in your Azure Active Directory (Azure AD) and configured into the App Service host.

Now, assign this service identity to a role in the required scope in your Service Bus resources.

1. Types of Authorization methods for different services:
   1. Service Bus authentication and authorization:
      1. Shared access policies: SAS Authentication
         1. Pros
            1. You have only one key to access, its simple, you can generate how many keys you prefer and use it in your code, you can also specify the type of access. Manage, send or listen the queues/topics.
            2. You can regenerate the keys to improve security
         2. Cons
            1. Its not safe enough, because you need to keep this key hardcoded.
            2. You can regenerate the keys to improve security, but you need to develop a way to connect to the service and generate a new key for you, its not simple to do it.
      2. Azure Active Directory (Azure AD) integration: Authenticate with managed identities
         1. Pros
            1. This is one of the best approach to authenticate and authorizate your apps. You need to generate a managed identity that will used to access you ServiceBus with no recycling password or secret concerns. You can leave to azure to manage it. Indicated for services like Azure WebApps, Functions, Virtual Machines, etc..
         2. Cons
            1. Not all services allow using managed identities.
      3. Azure Active Directory (Azure AD) integration: Authenticate from an application
         1. Pros:
            1. This is a common way to authenticate On-premise apps to your service bus or stora account.
         2. Cons:
            1. The main con is that you need to keep your app secret key stored in your code, even if you aren't hardcoding it, youll need to use it to authenticate.

1. Storage Account authentication and authorization:
   1. Shared Key (storage account key)
      1. Pros
         1. Same as SAS auth for ServiceBus, the storage account key has only one key and its simple for authenticate.
      2. Cons
         1. Its not safe enough, because you need to keep this key hardcoded.
         2. You can regenerate the keys to improve security, but you need to develop a way to connect to the service and generate a new key for you, its not simple to do it.

1. Shared access signature (SAS)
   1. Pros
      1. This is a handy way to share files and blobs as an one time access mode. You can generate a SAS URL from one specific blob and give this URL to someone else to download that file.
      2. An use case is when you want to copy a content from one storage account to another storage account. You can achieve it using Azcopy with SAS Auth.
   2. Cons
      1. The URL has a date to expires. You can choose for a day, a week, a month or even an year but, one day, it will expires. To avoid it, your application must has a routine to generate SAS tokens regularly.
2. Azure Active Directory (Azure AD) integration: Authenticate from an application
   1. Pros:
      1. This is a common way to authenticate On-premise apps to your service bus or stora account.
   2. Cons:
      1. The main con is that you need to keep your app secret key stored in your code, even if you aren't hardcoding it, youll need to use it to authenticate.
3. Azure Active Directory (Azure AD) integration: Authenticate with managed identities
   1. Pros
      1. This is one of the best approach to authenticate and authorizate your apps. You need to generate a managed identity that will used to access you ServiceBus with no recycling password or secret concerns. You can leave to azure to manage it. Indicated for services like Azure WebApps, Functions, Virtual Machines, etc..
   2. Cons
      1. Not all services allow using managed identities.
4. On-premises Active Directory Domain Services (preview)
   1. Pros
      1. This is useful for storageAccount SMB file shares when youre doing a lift and shift migration to the azure.
   2. Cons
      1. It only works for Storage Account SMB file share.
5. Anonymous public read access
   1. Pros
      1. This is perfect to store public access documentation or general data type, because you don’t need to autenticate, if you need to store and share something to the internet you can use the pubic access.
      2. This is very good to store website static public content.
   2. Cons
      1. There is no authentication, all the data will be available for reading through the internet.

1. References:
   1. Authorizing access to data in Azure Storage: <https://docs.microsoft.com/en-us/azure/storage/common/storage-auth?toc=/azure/storage/blobs/toc.json>
   2. Service Bus authentication and authorization: <https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-authentication-and-authorization#shared-access-signature>