Shared Access Signatures (SAS) in Azure allow you to grant limited access to resources in your Azure Storage account without exposing your account key. SAS tokens are particularly useful for scenarios where you need to provide access to Azure Blob Storage resources for a specific period, with restricted permissions.

Here’s a guide on how SAS works in a .NET application and how you can use it:

**How SAS Works**

1. **Generate SAS Token:**
   * A SAS token is generated to grant temporary access to resources. It includes a signature that is created using your storage account key and specifies the permissions and expiration time.
2. **Provide Access:**
   * Share the SAS token with the client or application that needs access. The client uses this token to access the Azure Storage resource.
3. **Access Control:**
   * The SAS token defines what actions can be performed (e.g., read, write) and how long the access is valid. It is important to manage the token’s expiration and permissions carefully.

**Generating SAS Token in .NET**

To generate a SAS token in a .NET application, you can use the Azure.Storage.Blobs library. Here’s an example of how to generate a SAS token for a blob container:

**Step 1: Install NuGet Package**

shell

Copy code

Install-Package Azure.Storage.Blobs

**Step 2: Generate SAS Token**

Here’s how to generate a SAS token for a blob container:

csharp

Copy code

using Azure.Storage.Blobs;

using Azure.Storage.Blobs.Models;

using Azure.Storage.Sas;

using System;

public class SasTokenGenerator

{

private readonly string \_storageAccountName = "your-storage-account-name";

private readonly string \_storageAccountKey = "your-storage-account-key";

private readonly string \_containerName = "your-container-name";

public string GenerateSasToken()

{

// Create a BlobServiceClient object to interact with the blob service

var blobServiceClient = new BlobServiceClient(new Uri($"https://{\_storageAccountName}.blob.core.windows.net"), new StorageSharedKeyCredential(\_storageAccountName, \_storageAccountKey));

// Create a BlobContainerClient object to interact with the container

var blobContainerClient = blobServiceClient.GetBlobContainerClient(\_containerName);

// Define the SAS token parameters

var blobSasBuilder = new BlobSasBuilder

{

BlobContainerName = \_containerName,

Resource = "c", // "c" for container, "b" for blob

ExpiresOn = DateTimeOffset.UtcNow.AddHours(1) // Set the expiration time

};

// Set permissions

blobSasBuilder.SetPermissions(BlobContainerSasPermissions.Parse("rl")); // "rl" for read and list

// Generate the SAS token

string sasToken = blobSasBuilder.ToSasQueryParameters(new StorageSharedKeyCredential(\_storageAccountName, \_storageAccountKey)).ToString();

return sasToken;

}

}

**Using SAS Token in .NET**

Once you have the SAS token, you can use it to interact with Azure Blob Storage. Here’s an example of how to use the SAS token to upload a file to a blob container:

**Step 1: Install NuGet Package**

shell

Copy code

Install-Package Azure.Storage.Blobs

**Step 2: Upload a File Using SAS Token**

csharp

Copy code

using Azure.Storage.Blobs;

using System;

using System.IO;

using System.Threading.Tasks;

public class BlobUploader

{

public async Task UploadBlobAsync(string sasToken, string containerName, string blobName, string filePath)

{

string blobServiceEndpoint = $"https://your-storage-account-name.blob.core.windows.net";

// Create a BlobContainerClient object with SAS token

var blobContainerClient = new BlobContainerClient(new Uri($"{blobServiceEndpoint}/{containerName}?{sasToken}"));

// Get a reference to the blob

var blobClient = blobContainerClient.GetBlobClient(blobName);

// Upload the file

using (var fileStream = File.OpenRead(filePath))

{

await blobClient.UploadAsync(fileStream, true);

}

}

}

**Summary**

* **Generate SAS Token:** Create a SAS token with specified permissions and expiration using the BlobSasBuilder class.
* **Use SAS Token:** Use the SAS token to interact with Azure Storage resources by appending it to the resource URL.

This approach allows you to securely grant temporary access to Azure Storage resources without sharing your storage account key.