GUIA DE DOWNLOAD DO ARQUIVO IMBARQ VIA API

HISTÓRICO DE ALTERAÇÕES

Data de atualização	Versão	Informações
18/04/2022	1.0	Versão inicial
25/05/2022	1.1	Adicionados informações de Configuração de Proxy (item 2.1 Configuração Proxy); e necessidade de inclusão de campos adicionais para listar os arquivos (item > 2.4 Listar os Arquivos que estão no contêiner).

SUMÁRIO

1 OBJETIVO

Este Guia de Download do Arquivo IMBARQ via API visa auxiliar as equipes técnicas que desejam desenvolver a automação do consumo dos arquivos IMBARQ no diretório nuvem Azure Blob Storage por meio da API.

Esse documento é um exemplo de como fazer o download do arquivo IMBARQ no Azure Blob Storage utilizando o SasToken, no Ambiente de Certificação.

2 PASSO A PASSO

A seguir, estão os passos necessários para fazer o download.

Nesse exemplo foi utilizado:

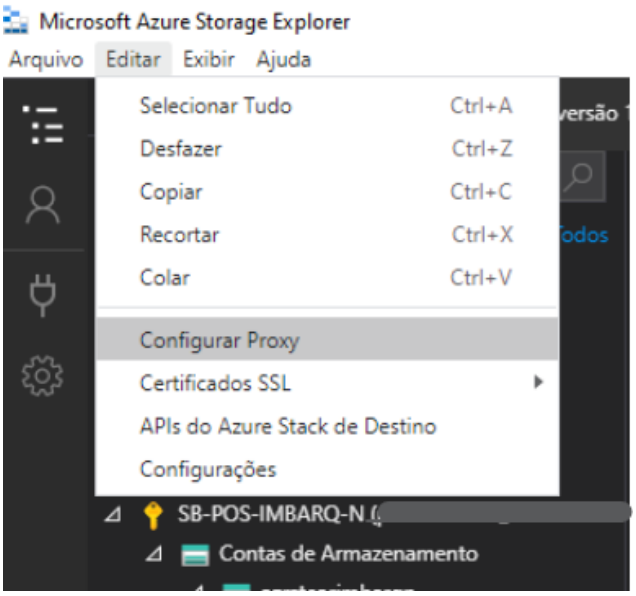
.Net Core 3.1

Azure.Storage.Blobs versão 12.11.0 - Biblioteca da Microsoft.

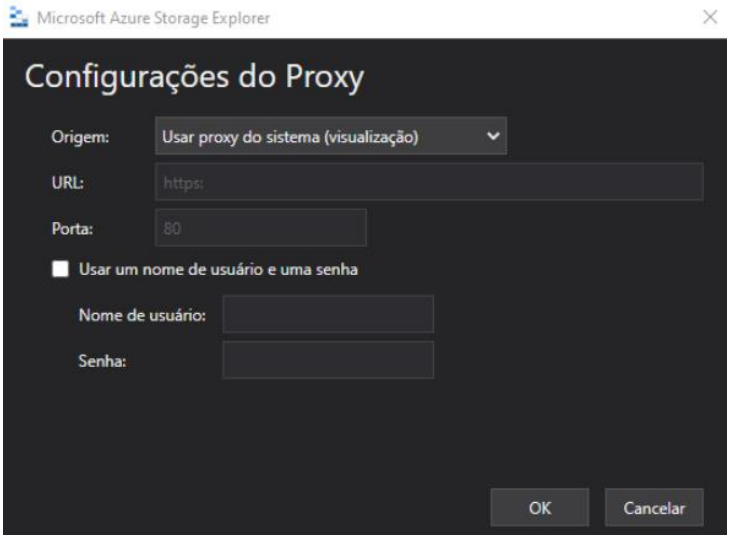
2.1 Configuração Proxy

Para utilização do Azure Explores é necessário realizar a seguinte configuração:

1. Com o Azure Storage Explorer aberto, no menu Editar procure o submenu Configurar Proxy:



2. Na opção Origem coloque e opção "Usar proxy do sistema(visualização)" e clique em OK:



2.2 Autenticação no API Gateway

Nesse primeiro passo é necessário que o participante possua uma Conta de Serviço e o Certificado Digital:

Parâmetros utilizados para o método **Autenticar_CAU**:

private static string url_auth = "https://api-listados-cert.b3.com.br/api/oauth/token";

private static string CATEGORY_ID = "39";

private static string grant_type = "client_credentials";

private static string client_id = "d532d356-8258-4610-9374-ab71591cbd82";

private static string client_secret = "8a9b98cd-970e-4ad6-918c-e891b9a00ee8";

Autenticação

```
static RetornoCau Autenticar_CAU()
{
    using (var httpClient = new HttpClient(GerarClientHandle()))
    {
        using (var request = new HttpRequestMessage(HttpMethod.Post, url_auth))
        {

            request.Headers.Add("CATEGORY_ID", CATEGORY_ID);
            HttpContent content = new FormUrlEncodedContent(new[]
            {
                new KeyValuePair<string, string>("grant_type", grant_type),
                new KeyValuePair<string, string>("client_id", client_id),
                new KeyValuePair<string, string>("client_secret", client_secret)
            });

            content.Headers.ContentType = new MediaTypeHeaderValue("application/x-www-form-urlencoded");
            request.Content = content;

            var response = httpClient.SendAsync(request).Result;

            if (response.StatusCode == System.Net.HttpStatusCode.OK)
            {
                var contentStram = response.Content.ReadAsStringAsync().Result;
                return JsonSerializer.Deserialize<RetornoCau>(contentStram);
            }
            else
            {
                Console.WriteLine($"Erro: {response.StatusCode.ToString()} - {response.RequestMessage}");
                return null;
            }
        }
    }
}
```

Método para inclusão do Certificado Digital (certificado e senha para o Participante 80980).

```
static HttpClientHandler GerarClientHandle()
```

```
    {
        HttpClientHandler handler = new HttpClientHandler();
        handler.ClientCertificateOptions = ClientCertificateOption.Manual;
        handler.SslProtocols = System.Security.Authentication.SslProtocols.Tls12;
        handler.ClientCertificates.Add(new X509Certificate2(@"C:\DEV\Certificado\s-80980-1\s-
80980-1.p12", "1234"));
        handler.ServerCertificateCustomValidationCallback = (httpRequestMessage, cert,
cetChain, policyErrors) => { return true; };
        return handler;
    }
```

Retorno

```
public class RetornoCau
{
    public string access_token { get; set; }
    public string token_type { get; set; }
    public string expires_in { get; set; }
    public string scope { get; set; }
}
```

2.3 Geração SAS TOKEN

Parâmetros utilizados para o método **GetSasToken**:

private static string url_sas_token = "https://api-listados-cert.b3.com.br/api/cip/v1/sas-tokens";

tokenCau: é esperado o access_token do retorno da Autenticação

```
static RetornoSasToken GetSasToken(string tokenCau)
    {
        using (var httpClient = new HttpClient(GerarClientHandle()))
        {
            using (var request = new HttpRequestMessage(HttpMethod.Get, url_sas_token))
            {
                request.Headers.Add("Authorization", string.Format("Bearer {0}", tokenCau));
                var response = httpClient.SendAsync(request).Result;

                if (response.StatusCode == System.Net.HttpStatusCode.OK)
                {
                    var contentString = response.Content.ReadAsStringAsync().Result;
                    return JsonSerializer.Deserialize<RetornoData>(contentString).data;
                }
                else
                {
                    Console.WriteLine($"Erro: {response.StatusCode.ToString()} -
{response.RequestMessage}");
```

```
                return null;
            }
        }
    }
}
```

Retorno

```
public class RetornoSasToken
{
    public Uri sasToken { get; set; }
}
```

SAS TOKEN

O SAS TOKEN tem uma validade de 1 hora, durante esse período não se faz necessário os passos 1 e 2 para pedir um novo SAS TOKEN.

2.4 Listar os Arquivos que estão no contêiner

Para conectar ao contêiner e listar os arquivos, utilizamos o método BlobContainerClient disponível na biblioteca Azure.Storage.Blobs.

Parâmetros utilizados para o método **ListarBlobs**:

sasToken - é necessário informar o SASTOKEN gerado.

Parâmetros adicionais - caso a listagem dos arquivos seja feita sem o SDK do Azure, com Postman ou outro método que usa HTTP, é necessário adicionar os parâmetros comp e restype, conforme abaixo:

comp = list

restype = container

Para mais informações, acessar a documentação oficial do Azure: https://docs.microsoft.com/en-us/rest/api/storageservices/list-blobs.

List Blobs (REST API) - Azure Storage

The List Blobs operation returns a list of the blobs under the specified container.

```
static List<Arquivo> ListarBlobs(Uri sasToken)
    {
        var Resultado = new List<Arquivo>();
        try
        {
            BlobContainerClient containerClient = new BlobContainerClient(sasToken);

            foreach (var blobItem in containerClient.GetBlobs())
            {
                Console.WriteLine($"Nome do Arquivo {blobItem.Name}");
                Resultado.Add(new Arquivo() { nomeArquivo = blobItem.Name, tipoArquivo =
blobItem.Properties.ContentType });
            }

            return Resultado;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Erro: {ex.Message}");
            return null;
        }
    }
```

Retorno

```
        public class Arquivo
{
    public string nomeArquivo { get; set; }
    public string tipoArquivo { get; set; }
```

2.5 Download do Arquivo

Para listar os arquivos, utilizamos o método BlobContainerClient disponível na biblioteca Azure.Storage.Blobs.

Para conectar ao contêiner utilizamos o método BlobContainerClient e para fazer o download do arquivo utilizamos o GetBlobCliente passando no nome do arquivo, ambos disponíveis na biblioteca Azure.Storage.Blobs.

Parâmetros utilizados para o método **DownloadFileBlob**:

sasToken - é necessário informar o sastoken gerado.

fileName - é o nome do arquivo que irá fazer o download

```
        static void DownloadFileBlob(Uri sasToken, string fileName)
{
    var Resultado = new List<string>();
    var path = $"C:\\DEV\\{fileName}";
    try
    {
        BlobContainerClient containerClient = new BlobContainerClient(sasToken);

        var blobClient = containerClient.GetBlobClient(fileName);

        using (var file = new FileStream(path, FileMode.Create))
        {
            blobClient.DownloadTo(file);
        }

    }
    catch (Exception ex)
    {
        Resultado.Add(ex.Message);
    }
}
```