Exemplo Java - Como criar uma "aplicação" para download do arquivo do blobStorage utilizando SASTOKEN

Esse documento é um exemplo de como fazer download de um documento no BlobStorage utilizando o SasToken, no Ambiente de Certificação.

Nesse exemplo foi utilizado:

Java

Azure.Storage.Blobs versão 12.11.0 - Biblioteca da Microsoft

Abaixo estão os passos necessários para fazer o download

1 - Autenticação no API Gatway

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

```
    private String autenticarCAU() {

                HttpEntity<?> entity = new HttpEntity<Object>(this.body.createBody(), this.header.
createHttpHeaderCAU());
                ResponseEntity<CAU> response = this.restTemplate.exchange(endpointCAU, HttpMethod.POST, entity,
CAU.class);
                System.out.println(response.getBody().getAccess_token());
                return response.getBody().getAccess_token();
        }
```

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

```
@Component
public class Header {

	@Value(value = "${categoryid}")
	private String categoryid;
	@Value(value = "${clientkeys}")
	private String clientkeys;

	@Bean
	public MultiValueMap<String, String> createHttpHeaderCAU() {

		MultiValueMap<String, String> header = new LinkedMultiValueMap<>();
		header.add("CATEGORY_ID", categoryid);
		header.add("X-SSL-Client-Cert-Subject", clientkeys);

		return header;
	}

	public MultiValueMap<String, String> createHttpHeaderSAS(String cauToken) {

		MultiValueMap<String, String> header = new LinkedMultiValueMap<>();
		header.add("Authorization", "Bearer "+ cauToken);

		return header;
	}
}
```

Retorno

```
    public class CAU {

        String access_token;
        String token_type;
        String expires_in;
        String scope;

        public String getAccess_token() {
                return access_token;
        }
        public void setAccess_token(String access_token) {
                this.access_token = access_token;
        }
        public String getToken_type() {
                return token_type;
        }
        public void setToken_type(String token_type) {
                this.token_type = token_type;
        }
        public String getExpires_in() {
                return expires_in;
        }
        public void setExpires_in(String expires_in) {
                this.expires_in = expires_in;
        }
        public String getScope() {
                return scope;
        }
        public void setScope(String scope) {
                this.scope = scope;
        }

}
```

2 - 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

```
         private String getSasToken(String cauToken) {

                HttpEntity<?> entity = new HttpEntity<Object>(null, this.header.createHttpHeaderSAS(cauToken));
                ResponseEntity<SASToken> response = this.restTemplate.exchange(endpointSAS, HttpMethod.GET,
entity,
                                SASToken.class);

                System.out.println(response.getBody().getData().getSasToken());
                return response.getBody().getData().getSasToken();

        }
```

Retorno

```
   public class SASToken {

        Data data;
        Links links;

        public Data getData() {
                return data;
        }
        public void setData(Data data) {
                this.data = data;
        }
        public Links getLinks() {
                return links;
        }
        public void setLinks(Links links) {
                this.links = links;
        }
}
```

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

3 - 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.

```
        private void listarBlobs(String sasToken) {

                BlobContainerClient blobContainerClient = new BlobContainerClientBuilder()
                                .endpoint(sasToken)
                                .buildClient();

        List<String> list = new ArrayList<>();
        PagedIterable<BlobItem> listBlobs = blobContainerClient.listBlobs();
        listBlobs.forEach(item -> list.add(item.getName()));
        for (String string : list) {
            System.out.println(string);
        }
        }
```

Retorno

```
@SpringBootApplication
@Configuration
@ComponentScan("br.com.b3")
public class Application implements CommandLineRunner {

	@Autowired
	RestTemplate restTemplate;
	@Autowired
	Header header;
	@Autowired
	Body body;

	@Value(value = "${endpointCAU}")
	private String endpointCAU;

	@Value(value = "${endpointSAS}")
	private String endpointSAS;

	public static void main(String[] args) {
		SpringApplication.run(Application.class, args);

	}

	@Override
	public void run(String... args) throws Exception {

		String cauToken = autenticarCAU();
		String sasToken = getSasToken(cauToken);
		listarBlobs(sasToken);
	}

	private void listarBlobs(String sasToken) {

		BlobContainerClient blobContainerClient = new BlobContainerClientBuilder()
				.endpoint(sasToken)
				.buildClient();

	List<String> list = new ArrayList<>();
	PagedIterable<BlobItem> listBlobs = blobContainerClient.listBlobs();
	listBlobs.forEach(item -> list.add(item.getName()));
	for (String string : list) {
	    System.out.println(string);
	}
	}

	private String getSasToken(String cauToken) {

		HttpEntity<?> entity = new HttpEntity<Object>(null, this.header.createHttpHeaderSAS(cauToken));
		ResponseEntity<SASToken> response = this.restTemplate.exchange(endpointSAS, HttpMethod.GET,
entity,
				SASToken.class);

		System.out.println(response.getBody().getData().getSasToken());
		return response.getBody().getData().getSasToken();

	}

	private String autenticarCAU() {

		HttpEntity<?> entity = new HttpEntity<Object>(this.body.createBody(), this.header.
createHttpHeaderCAU());
		ResponseEntity<CAU> response = this.restTemplate.exchange(endpointCAU, HttpMethod.POST, entity,
CAU.class);
		System.out.println(response.getBody().getAccess_token());
		return response.getBody().getAccess_token();
	}

}
```

4 - 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

Fonte:

Sample in Java, (imb-sample.zip)