API AUTHENTICATION

All API requests need to be authenticated through the Authorization header. The API offers the following authentication methods:

- HTTP Basic authentication
- API Keys

Select your preferred method to suit your current tech stack and security requirement level. Many of these methods are vulnerable to man-in-the-middle attacks, so it is recommended to combine them with other security mechanisms such as an encrypted connection or SSL.

BASIC

Basic authentication works by sending a username and password in every API request. Typically, this method is used in situations when the API key is not available. For example, API methods generating API keys could be authenticated with Basic.

Basic authentication is the least recommended method as it is still simple to decode encrypted credentials back to their original values. Please refer to the HTTP Authentication resource to see how to restrict access to your server with Basic authentication.

Here are some key facts about this method:

- Built into HTTP protocol itself
- Credentials should be encoded in a Base64 format (e.g. with the RFC2045-MIME variant) and separated by a colon :

- Encoded credentials are added to the header after "`Basic`"

Example - HTTP client

When using any of the API client libraries you don't have to manually encode the credentials. You only need to specify the username and password when creating an instance of a client object.

CURL

```
curl -L -g -X GET 'https://{baseUrl}/sms/2/text/advanced' /
-H 'Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ=='
```

JAVA

```
String auth = username + ":" + password;
String encoded = Base64.getEncoder().encodeToString(auth.getBytes("UTF-8"));

String messageText = "This is test message sent using OkHttpClient and Basic auth";
String jsonBody = String.format(
        "{\"messages\":[{\"from\":\"%s\",\"destinations\":[{\"to\":\"%s\"}],\"text\":\"%s\"}]}",
        SENDER, PHONE_NUMBER, messageText);

RequestBody requestBody = RequestBody.create(MediaType.parse("application/json; charset=utf-8"), jsonBody);

Request request = new Request.Builder()
        .url(BASE_URL + "/sms/2/text/advanced")
        .addHeader("Authorization", "Basic " + encoded)
        .addHeader("Content-Type", "application/json")
        .addHeader("Accept", "application/json")
        .post(requestBody)
        .build();

OkHttpClient httpClient = new OkHttpClient().newBuilder().build();
Response response = httpClient.newCall(request).execute();
System.out.println(response.body().string());
```

C#

```
string username = "USERNAME";
string password = "PASSWORD";
string concatenated = $":{password}";
string encoded = Convert.ToBase64String(System.Text.Encoding.UTF8.GetBytes(concatenated));

HttpClient client = new HttpClient();
client.BaseAddress = new Uri("BASE_URL");
client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Basic", encoded);
client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
```

Example - API client library

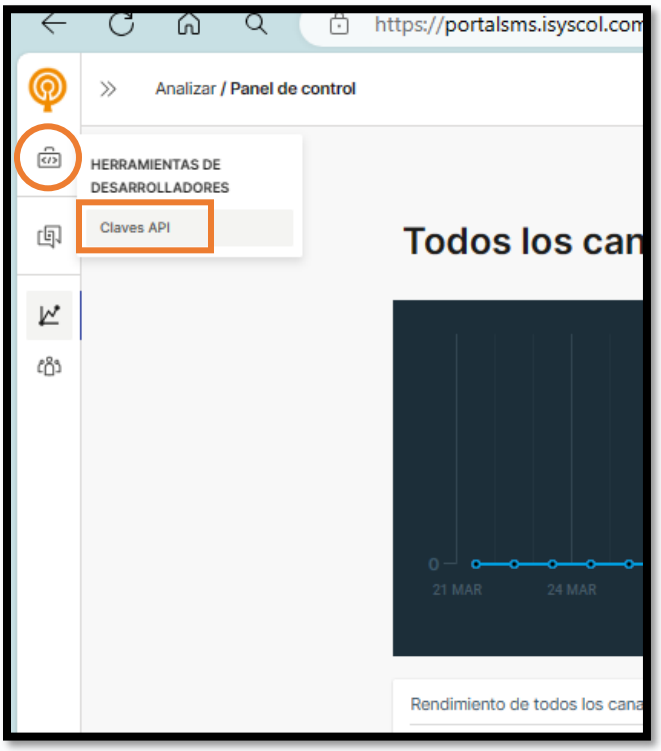
When using any of the API client libraries you do not have to manually encode the credentials data like mentioned above. You only need to specify the username and password when creating an instance of a client object as shown in the example below.

C#

```
var configuration = new Configuration()
{
    BasePath = "BASE_URL",
    Username = "USERNAME",
    Password = "PASSWORD"
};
```

API KEY HEADER

An API key is an access token that a client provides when making API calls. It's a simple way to secure access and thus the most popular authentication method used with REST APIs. The key can be sent in the query string or as a request header. You can generate keys and manage the existing ones through the Web portal.



Here are some key facts about this method:

- API keys can be generated by calling the dedicated API method
- Keys can be revoked at any time which is useful when separating the API access rights across multiple applications or use cases
- Lost API keys are easily retrievable
- API keys have a predefined expiry date to eventually become invalid

Example - HTTP client

The examples below show how to specify the API Key authentication when using client libraries.

CURL

```
curl -L -g -X GET 'https://{baseUrl}/sms/2/text/advanced' /


-H 'Authorization: App 003026abc133714df1834b8638bb496e-8f4b3d9a-e931-478d-a994-28a725159ab9'
```

JAVA

```
Request request = new Request.Builder()
        .url("BASE_URL" + "/sms/2/text/advanced")
        .addHeader("Authorization", "App " + apiKey)
        .addHeader("Content-Type", "application/json")
        .addHeader("Accept", "application/json")
        .post(requestBody)
        .build();
```

C#

```


HttpClient client = new HttpClient();
client.BaseAddress = new Uri("BASE_URL");

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("App", "API_KEY");
client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
```

Example - API client library

The examples below show how to prepare an HTTP request using API Key authentication. Note this request is much simpler than using a basic authentication request.

JAVA

```
ApiClient apiClient = ApiClient.forApiKey(ApiKey.from(API_KEY))
    .withBaseUrl(BaseUrl.from(BASE_URL))
    .build();
```

C#

```
var configuration = new Configuration()
{
    BasePath = "BASE_URL",
    ApiKeyPrefix = "App",
    ApiKey = "API_KEY"
};
```