# Programming the Power BI Service API

Now it's time to write a complete C# program that calls the Power BI Service API. First, the program must implement an authentication flow to call to Azure AD and obtain an access token for the Power BI Service API. Second, the application must transmit the access token along with any HTTP request sent to the Power BI Service API. Let's start by examining the entire program at once and, after that, we'll walk through smaller sections of this code.

Now let's move on to the *ExecuteGetRequest* function which uses an *HttpClient* object to execute an HTTP GET operation. The main point to see here is that this function adds the *Authorization* header to each requests and sets the value for this header to a string that combines the word "Bearer" together with a space and an access token returned from the *GetAccessToken* function. As long as the call to *client.SendAsync* returns a successful HTTP status code, the *ExecuteGetRequest* function returns the content from the HTTP GET operation as a string.

static string ExecuteGetRequest(string restUrl, string accessToken) {

HttpClient client = new HttpClient();

HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Get, restUrl);

request.Headers.Add("Authorization", "Bearer " + accessToken);

request.Headers.Add("Accept", "application/json;odata.metadata=minimal");

HttpResponseMessage response = client.SendAsync(request).Result;

if (response.StatusCode != HttpStatusCode.OK) {

throw new ApplicationException("Error occured calling the Power BI Servide API");

}

return response.Content.ReadAsStringAsync().Result;

}

When can now move on to the last function named *Main*. When the *Main* function begins to execute, it calls *ExecuteGetRequest* and passes the REST URL required by the Power BI Service API to retrieve the reports in the current users personal workspace.

string restUrl = "https://api.powerbi.com/v1.0/myorg/reports/";

var json = ExecuteGetRequest(restUrl);

The call to *ExecuteGetRequest* triggers a call to *GetAccessToken* which begins the authentication flow. The user should be prompted to log in by entering a user name and password. If delegated permissions have not yet been granted, the user will be prompted with the consent dialog. Once the user has completed the interactive login, the call to GetAccessToken returns an access token back to the *ExecuteGetRequest* function which then passes the access token in the *Authorization* header when it calls to the Power BI Service. Figure 3.11 shows an example of using the popular Fiddler utility to inspect a call to the Power BI Service and see the access token that's being transmitted in the *Authorization* header.

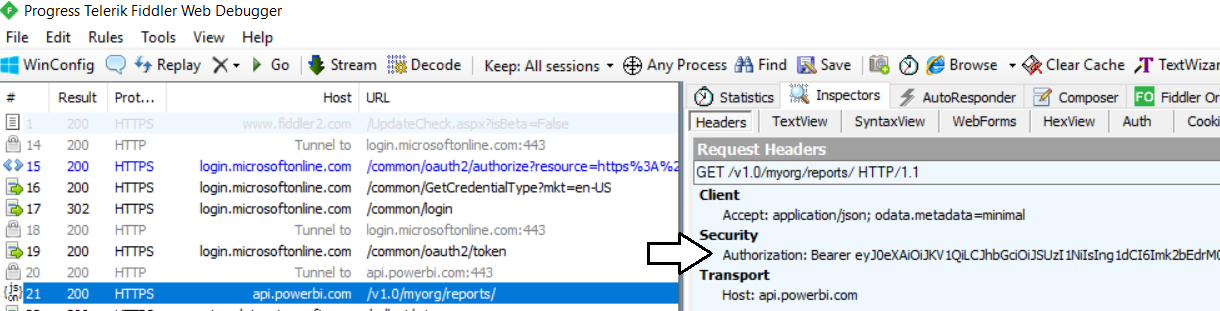


Figure 3.11: All calls to the Power BI Service API must pass an access token in the Authorization header

At this point, the call to *ExecuteGetRequest* returns back to *Main* with a string value containing the JSON returned from the Power Service API with the data for the reports in the user's personal workspace. The Power BI Service usually returns JSON results in a standard ODATA format as shown in Figure 3.12.



Figure 3.12: This is an example of the JSON format returned by the Power BI Service API.

The final step now is to convert the JSON string value into strongly-typed objects to simply the code. This can be accomplished by defining the following two classes in C# and using them in a call to *JsonConvert.DeserializeObject*.

public class Report {

public string id { get; set; }

public string name { get; set; }

public string webUrl { get; set; }

public string embedUrl { get; set; }

public bool isOwnedByMe { get; set; }

public string datasetId { get; set; }

}

public class ReportCollection {

public List<Report> value { get; set; }

}

The call to *JsonConvert.DeserializeObject* returns an object with a *value* property containing a collection of *Report* objects that you can enumerate using a C# foreach loop. This makes it possible to display the names of each report to the console window.

static void Main() {

var json = ExecuteGetRequest("https://api.powerbi.com/v1.0/myorg/reports/");

ReportCollection reports = JsonConvert.DeserializeObject<ReportCollection>(json);

foreach (Report report in reports.value) {

Console.WriteLine(report.name);

}

}

### Developing with the Power BI .NET SDK

Power BI Client

All that security work is behind us. Now all we have to think about is getting a PowerBIClient object and leveraging the API

#### Understanding the Difference Between User APIs and Admin APIs

#### Calling the Power BI Service API as a Service Principle

#### Generating Embed Tokens for Power BI Resources

## Retrieving the Data for Power BI Embedding

### Embedding Reports

Zzzz

class ReportEmbeddingData {

public string reportId;

public string reportName;

public string embedUrl;

public string accessToken;

}

zzz