Power BI API Usage

15/03/2018

White Paper

Relationship: Alcoa-Arconic

Aniket Sarkar

MSBI, Microsoft Technologies

[aniket.sarkar@tcs.com](mailto:aniket.sarkar@tcs.com)



|  |
| --- |
| **Confidentiality Statement** |
| Include the confidentiality statement within the box provided. This has to be legally approved  **Confidentiality and Non-Disclosure Notice**  The information contained in this document is confidential and proprietary to TATA Consultancy Services. This information may not be disclosed, duplicated or used for any other purposes. The information contained in this document may not be released in whole or in part outside TCS for any purpose without the express written permission of TATA Consultancy Services. |

|  |
| --- |
| **Tata Code of Conduct** |
| We, in our dealings, are self-regulated by a Code of Conduct as enshrined in the Tata Code of Conduct. We request your support in helping us adhere to the Code in letter and spirit. We request that any violation or potential violation of the Code by any person be promptly brought to the notice of the Local Ethics Counselor or the Principal Ethics Counselor or the CEO of TCS. All communication received in this regard will be treated and kept as confidential. |

**Table of Content**

[Abstract 4](#_Toc509564355)

[About the Author 5](#_Toc509564356)

[About the Domain 5](#_Toc509564357)

[1. RESTful Service 6](#_Toc509564358)

[2. Azure Active Directory and Azure App 6](#_Toc509564359)

[2.1 Step by Step App Registration 6](#_Toc509564360)

[2.1.1 Azure portal for App registrations can be reached through: https://azure.portal.com 6](#_Toc509564361)

[2.1.2 Go to Azure Data Catalog 🡪 App Registrations 🡪 New App Registration 7](#_Toc509564362)

[2.1.3 Provide the Display Name, Application Type as “Native”. 7](#_Toc509564363)

[2.1.4 Under “Required Permissions”, add the services which this App will use. This should contain the Windows Azure Active Directory and Power BI service. 8](#_Toc509564364)

[3. Prerequisites 8](#_Toc509564365)

[4. Azure Authentication 8](#_Toc509564366)

[5. RESTful methods of PowerBI API 10](#_Toc509564367)

[5.1 Check Workspace Name 10](#_Toc509564368)

[5.2 Get Reports and Dashboards Names 10](#_Toc509564369)

[5.3 Copy Workspace Contents 10](#_Toc509564370)

[5.4 Refresh Datasets 11](#_Toc509564371)

[6. Invoking PowerShell scripts through ISE 11](#_Toc509564372)

[7. Invoking PowerShell scripts 11](#_Toc509564373)

[7.1 C# method to execute PS1 files 11](#_Toc509564374)

[8. Conclusion 13](#_Toc509564375)

[9. Acknowledgements 14](#_Toc509564376)

[10. References 15](#_Toc509564377)

# Abstract

PowerBI is a cloud based dynamic reporting tool by Microsoft. Although activities and events on PowerBI datsets and workspaces can be done through the online portal, Microsoft has provided API for developers to perform the remote and/or bulk activities.

The Whitepaper describes the usage of API to copy datasets, reports and workspaces from one PowerBI workspace to another; and refresh datasets. The API is invoked via PowerShell scripts and the scripts are in turn invoked by user selection from Asp.Net web pages.

# About the Author

Aniket Sarkar has above eight years of experience in Microsoft technologies; specialized in BI space.

The prime technology stack includes latest versions of SQL, SSIS, SSAS, Tabular and SSRS.

He has worked with renowned clients like Humana (Healthcare) and Alcoa-Arconic (Resources).

He has provided unique and near optimal solutions to most of the customer’s analytical requirements and is continuing to improve.

He is engaged in learning secondary technologies like ASP.Net and PowerShell and scrutinize next gen technologies including Azure and PowerBI.

# About the Domain

Microsoft BI is a complete product suite for designing Business Analytics solutions with capability of providing various data modelling and reporting tools, having extensive features. The solution discussed in this document scope is built on the following components:

* Microsoft SQL Server 2012
* Microsoft SQL Server Integration Services 2012
* Microsoft SharePoint 2013
* Microsoft Office Excel 2016

# RESTful Service

REpresentational State Transfer (REST), or RESTful, web services provide interoperability between computer systems on the Internet. REST-compliant web services allow the requesting systems to access and manipulate textual representations of web resources by using a uniform and predefined set of stateless operations. A stateless protocol is a communications protocol in which no information is retained by either sender or receiver. PowerBI API uses REST methods to perform certain activities.

# Azure Active Directory and Azure App

PowerBI or any other Microsoft cloud service authenticates users via AAD (Azure Active Directory). To use any service of Azure, a specific “App” needs to be registered at Azure portal (portal.azure.com).

Two vital properties need to be configured within the registered app:

1. URLs which the app will access
2. AAD User IDs to be granted access to the URLs.

Once registered, Azure would provide a Client ID.

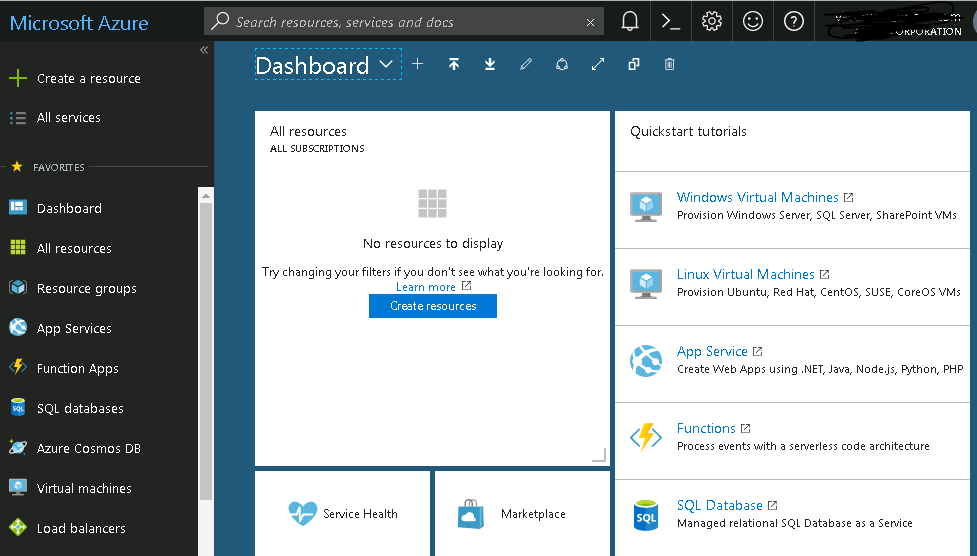
This App ID needs to be used in client applications from where any API is invoked.

The Azure App (Client ID) will act as a pivot router for any API method requests.

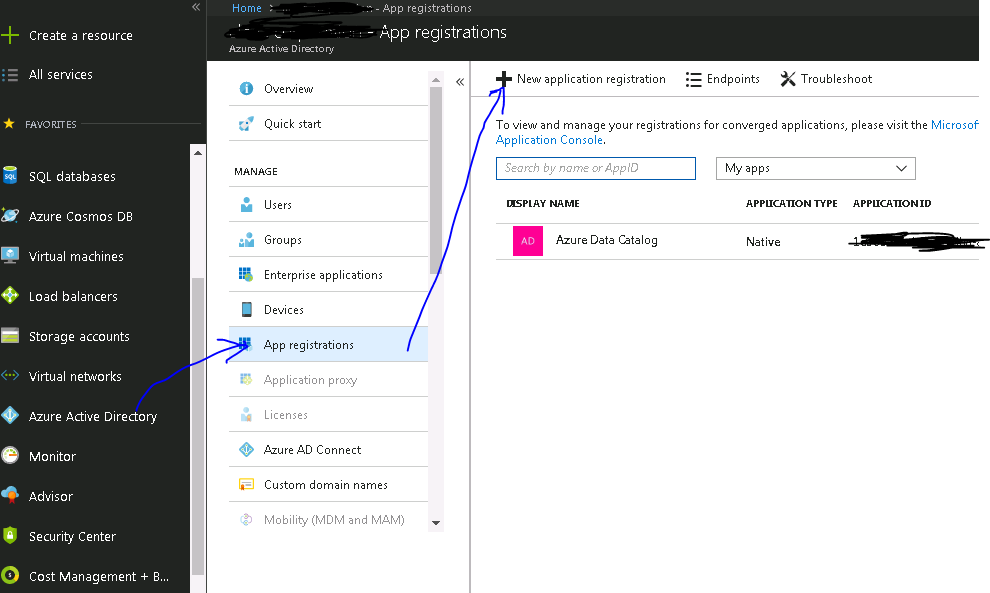
The app registration and Client ID can be considered as one time and permanent.

## Step by Step App Registration

### Azure portal for App registrations can be reached through: <https://azure.portal.com>



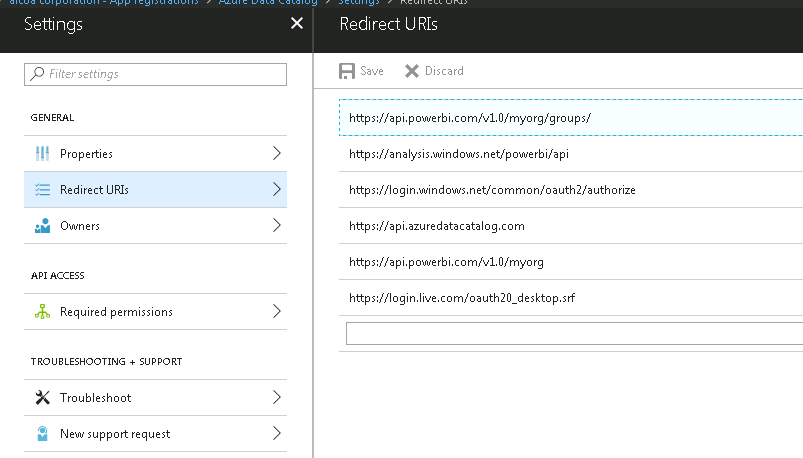
### Go to Azure Data Catalog 🡪 App Registrations 🡪 New App Registration



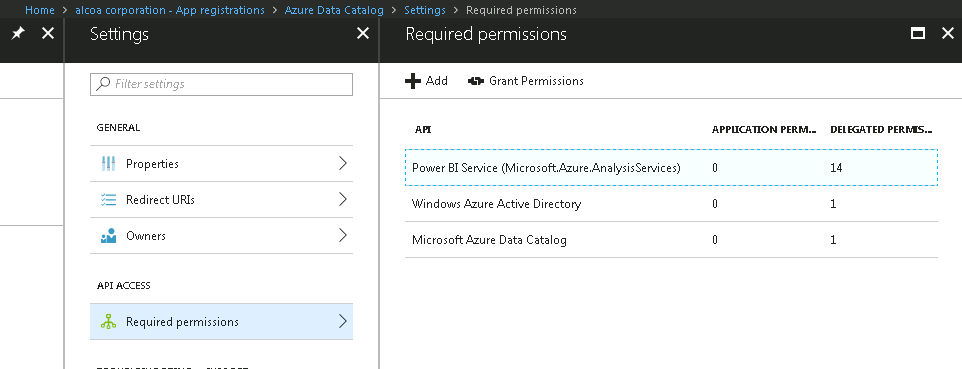
### Provide the Display Name, Application Type as “Native”.

Mention all API URLs which the app will access under “Redirect URI”. For this project, it should contain the PowerBI API URI, along with some genral Microsoft and Windows URIs.

Can add certain Azure AD accounts to the “Owners” section of this App settings.



### Under “Required Permissions”, add the services which this App will use. This should contain the Windows Azure Active Directory and Power BI service.



Once the App is created, it would generate an “Application ID” or “Client ID”. This Client ID can now be utilized by client applications.

# Prerequisites

In order to invoke the PowerBI related PowerShell scripts, the machine should have:

* PowerShell 3.0 or above
* Azure PowerShell Cmdlets installed

# Azure Authentication

Once the App is registered, the App’s Client ID should be used to authenticate a user against AAD.

To invoke an Azure or specifically PowerBI service via RESTful API, an Authorization Token is needed. The Authorization token is received once the client application is authenticated. To receive a valid Auth Token, the Client ID and an AAD credential is passed as a parameter to an authorization method. If valid, the invoking method will receive an Auth Token, which will be required while invoking any Rest method of the PowerBI API.

Before invoking the authorization method, the following Azure dlls need to be loaded, using Add-Type command:

Add-Type -Path 'C:\Program Files (x86)\Microsoft SDKs\Azure\PowerShell\ServiceManagement\Azure\Services\Microsoft.IdentityModel.Clients.ActiveDirectory.dll'

Add-Type -Path 'C:\Program Files (x86)\Microsoft SDKs\Azure\PowerShell\ServiceManagement\Azure\Services\Microsoft.IdentityModel.Clients.ActiveDirectory.WindowsForms.dll'

In case of a 32 bit machine, the Program Files (x86) would be replaced by Program Files.

A user credential needs to be created through commands:

$clientId = "1e06b58a-b864-4dba-a520-a3e92b0dd06a"

$user\_id = "test@test.com"

$password = ConvertTo-SecureString "xxxxxx" -AsPlainText –Force

$UserCredential = New-Object -TypeName Microsoft.IdentityModel.Clients.ActiveDirectory.UserCredential -ArgumentList ($user\_id, $password)

The Client ID which was generated through the App registration, should be defined within the script.

$clientId = "1e06324234b58a-b864-4d5546ba-a5s20-a3e06a"

The Authorization Token can be acquired by the following method.

function GetAuthToken

{

$adal = "${env:ProgramFiles(x86)}\Microsoft SDKs\Azure\PowerShell\ServiceManagement\Azure\Services\Microsoft.IdentityModel.Clients.ActiveDirectory.dll"

$adalforms = "${env:ProgramFiles(x86)}\Microsoft SDKs\Azure\PowerShell\ServiceManagement\Azure\Services\Microsoft.IdentityModel.Clients.ActiveDirectory.WindowsForms.dll"

[System.Reflection.Assembly]::LoadFrom($adal) | Out-Null

[System.Reflection.Assembly]::LoadFrom($adalforms) | Out-Null

$redirectUri = "https://login.live.com/oauth20\_desktop.srf"

$resourceAppIdURI = "https://analysis.windows.net/powerbi/api"

$authority = "https://login.windows.net/common/oauth2/authorize";

$authContext = New-Object -TypeName Microsoft.IdentityModel.Clients.ActiveDirectory.AuthenticationContext -ArgumentList ($authority)

$authResult = $authContext.AcquireToken($resourceAppIdURI, $clientId, $UserCredential)

return $authResult

}

**$token** = GetAuthToken

# RESTful methods of PowerBI API

The RESTful methods can be invoked via PowerShell scripts or through C# code.

All examples below use PowerShell script to demonstrate the capabilities of PowerBI API.

All scripts, first get an authorization token and then carry on its distinctive activity.

A generalized example of RESTful method invocation would be:

$response = (Invoke-RestMethod -Uri $uri –Headers $auth\_header -ContentType 'application/json' –Method GET).value

It accepts parameters:

* URI: for PowerBI service, the URI would be <https://api.powerbi.com/v1.0/myorg/groups/>
* Headers: a header object to be created using the authorization token already received

$auth\_header = @{

'Authorization'=$token.CreateAuthorizationHeader()

}

* Content Type: would always be application/json
* Method: Get, Post or Delete

## Check Workspace Name

This PowerShell script gets the names of all PowerBI workspaces to which the Authenticated ID has access to. The list of workspaces are received in an object.

The list of workspaces is checked against a particular workspace name which is provided as a parameter.

The error (if any) is written down into an output text file.



## Get Reports and Dashboards Names

This PowerShell script gets list of Reports and Dashboards within a specified Workspace (passed as parameter). The lists are then written down into two separate files, the path of whose are provided as parameters.



## Copy Workspace Contents

The PowerShell scripts copies specified objects from one workspace to another. The specific objects (reports and dashboards) are read from two separate text files, which is in turn provided as parameters.

The two scripts are similar except that one copies over everything to the destination workspace even if the destination has the same objects; while the other one ignores the copy activity for the objects for which it finds a match.

## Refresh Datasets

The PowerShell script can refresh mentioned PowerBI datasets. If the dataset connects to a cloud based service, no “gateway” is required. If the datset connects to an on-premise datasource, specific “gateway” should be setup so that the cloud based PowerBI can access the on-premise source.



# Invoking PowerShell scripts through ISE

Microsoft has provided an Integrated Scripting Environment, where developers can test out their PowerShell scripts before actual implementation.

The ISE has predefined libraries loaded and has intellisense which help developers during development of the script. Various properties of an object can be browsed and the required one can be selected.

In order to execute a certain PowerShell script file (.ps1), the full path of the script file (including file name) should be mentioned in the console. If the file expects one or more parameters, the parameters can be passed after the file path name, within double quotes. Multiple parameters should be separated with a space.

# Invoking PowerShell scripts

The idea is to create a web application, which will be intuitive and friendly to use. The high level functionality of the web application would be to copy workspace contents from one to another. (Migration of workspaces); and refresh datsets of any given workspace.

The web application can be smart enough to accept the source workspace name, list down its contents, for example, all its datasets/related reports and dashboards. The page should provide the capability to select one or more contents and through a button push, the chosen objects would be copied over to the given destination workspace.

## C# method to execute PS1 files

The ideas described above are standard web page development scenario; which would not be technically described here. Rather, only the execution of PowerShell script files through a page button click shall be detailed.

It is advised to keep all scripts in a single directory within the web server which hosts the web application.

Nuget namespaces

The following dll needs to be included within the .Net project:

System.Management.Automation

Once added under References of the Solution, need to use the namespaces:

using System.Management.Automation;

using System.Management.Automation.Runspaces;

The following method provides the detail on how to execute a PowerShell script file:

private static void RunPowershellScript(string scriptFile, List<string> parameters)

{

// Validate parameters

if (string.IsNullOrEmpty(scriptFile)) { throw new ArgumentNullException("scriptFile"); }

if (parameters == null) { throw new ArgumentNullException("parameters"); }

RunspaceConfiguration runspaceConfiguration = RunspaceConfiguration.Create();

using (Runspace runspace = RunspaceFactory.CreateRunspace(runspaceConfiguration))

{

runspace.Open();

RunspaceInvoke scriptInvoker = new RunspaceInvoke(runspace);

scriptInvoker.Invoke("Set-ExecutionPolicy Unrestricted");

Pipeline pipeline = runspace.CreatePipeline();

Command scriptCommand = new Command(scriptFile);

Collection<CommandParameter> commandParameters = new Collection<CommandParameter>();

foreach (string scriptParameter in parameters)

{

CommandParameter commandParm = new CommandParameter(null, scriptParameter);

commandParameters.Add(commandParm);

scriptCommand.Parameters.Add(commandParm);

}

pipeline.Commands.Add(scriptCommand);

Collection<PSObject> psObjects;

psObjects = pipeline.Invoke();

}

}

The method can be called in this manner:

RunPowershellScript(SourceWSValidateScriptFile, new List<string>(new string[] { SrcWS, SourceWSLogFile }));

**7.2 Error Handling and Logging**

Once C# code executes a PS1 file, it works synchronously. Compiler control would wait for the next line to be executed till the file completes its set of instructions.

However, the PS1 file cannot communicate with C# code. Any response provided by the script cannot be passed on to the invoking C# method.

So, any log or error messages cannot be directly passed to the calling C# method.

To overcome this situation, all logging activities has to be carried out from within the script into a text file.

Upon execution of the script, the C# code can then read from the log / error text file and display it over the page.

If the error file contains text greater than length one, it signifies that an error has occurred, and based on that, next set of activities of the web application can be defined.

# Conclusion

Utilizing the capabilities of PowerBI API, developers can create tools for PowerBI admins.

Assuming a web based tool, admins can perform migrations of PowerBI components from Dev environment (workspace) to Prod environment.

Admins can also refresh datasets on demand, which would make the PowerBI report ready for business adhoc.

Once more functionalities are added by Microsoft within the API, those functionalities can also be incorporated within the tool.

# Acknowledgements

This technical achievement would have been impossible without the continuous help from the team lead “Partha De” (233118).

The idea behind the web based tool to accomplish remote activities through API was the brain child of our customer, so a special thanks to him.

# References

<https://docs.microsoft.com/en-us/power-bi/developer/rest-api-reference>

<https://powerbi.microsoft.com/en-us/blog/duplicate-workspaces-using-the-power-bi-rest-apis-a-step-by-step-tutorial/>

**Thank You**

**Contact**

For more information, contact [**gsl.cdsfiodg@tcs.com**](mailto:gsl.cdsfiodg@tcs.com)(Email Id of ISU)

**About Tata Consultancy Services (TCS)**

Tata Consultancy Services is an IT services, consulting and business solutions organization that delivers real results to global business, ensuring a level of certainty no other firm can match. TCS offers a consulting-led, integrated portfolio of IT and IT-enabled infrastructure, engineering and assurance services. This is delivered through its unique Global Network Delivery ModelTM, recognized as the benchmark of excellence in software development. A part of the Tata Group, India’s largest industrial conglomerate, TCS has a global footprint and is listed on the National Stock Exchange and Bombay Stock Exchange in India.

For more information, visit us at **www.tcs.com**.

**IT Services**

**Business Solutions**

**Consulting**

All content / information present here is the exclusive property of Tata Consultancy Services Limited (TCS). The content / information contained here is correct at the time of publishing. No material from here may be copied, modified, reproduced, republished, uploaded, transmitted, posted or distributed in any form without prior written permission from TCS. Unauthorized use of the content / information appearing here may violate copyright, trademark and other applicable laws, and could result in criminal or civil penalties. **Copyright © 2011 Tata Consultancy Services Limited**