Power BI On-premises Gateway Cluster Monitoring (Power Platform API)

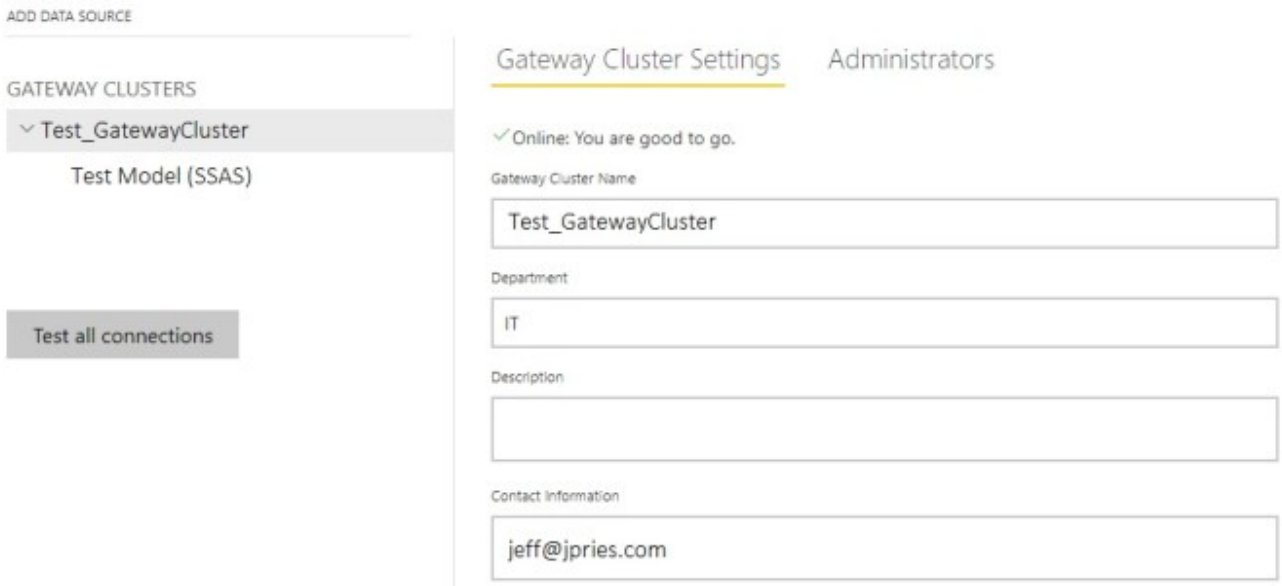
blog.jpries.com/2020/01/10/power-bi-on-premises-gateway-cluster-monitoring

jpries January 10, 2020

Failover clustering and load balancing was introduced to the Power BI On-premises (aka Enterprise Gateway) almost two years ago in the November, 2017 update of the software.

Recently having the need to add redundancy to a configuration of the gateway, I eagerly began researching how to best implement this configuration.

Upon successfully installing a second member to the cluster, it's interesting to note that there is absolutely no indication in the web interface that the cluster is now two members strong instead of one. Even more interesting is that this has been this way since it was implemented in 2017.



Pop Quiz: How many Gateway Agents are installed in this Cluster? (Did you guess 2? No? Why would you?)

So, how do we see how many agents are installed in our Gateway Cluster and which version of the software is installed on each? PowerShell to the rescue once again!

I'm not going to cover how to query the Gateway Cluster status via the OnPremisesDataGatewayHAMgmt.psm1 PowerShell module — Craig Porteous does an excellent job in his article, here and the full syntax of the module is described in this Microsoft article. Though, I will note a few gotchas I encountered with it.

- This module is a standalone .psm1 file located in the On-premises Gateway installation directory. It isn't a part of the usual Power BI Management cmdlets (at least not yet).
- The module requires that you have PowerShell 5 or greater — though note that PowerShell 6 Core won't work (as it uses a graphical prompt for authentication, which uses Windows Forms, which isn't included with PowerShell 6 Core).
- The PowerShell module can be run from any machine, though it will need access to the On-premises Gateway installation directory, as it will read some settings out of a .config file and reference a few of the .dll files located there.

Having a PowerShell module to gather information about the installed Gateway Agents is great, but I wanted something I could automate a little more that didn't necessarily need access to the original installation directory (though you could potentially just copy the couple of files referenced out of it to another machine for future executions).

Update 10/15/2019 — While putting this piece together, Microsoft announced that they have a suite of new PowerShell cmdlets available for the purpose of managing the On-premises gateway! They are still in Preview, but check them out here and here and read about Aaron Nelson's initial experience with them here. They do require PowerShell Core 6 (which is different than the normal Windows PowerShell built into modern versions of Windows), which is a bit of a nuisance. The PowerShell cmdlets do make use of a different API command than I use here (https://api.powerbi.com/v2.0/myorg/me/gatewayClusters) so I will likely do a follow-up post in the future using this new command.

For my solution, since I was already working with C# to query the Power BI API, I figured, "why not also query this management interface with C#?"

After looking through the OnPremisesDataGatewayHAMgmt.psm1 PowerShell module to learn how it worked its magic specifically what it queried and from where, I was able to work out exactly how to query for this information — and the good news was that while it wasn't exactly the normal Power BI API (it's the Power Platform API), it was very, very similar.

The PowerShell module performs 4 basic steps which will need to be emulated in the application:

Fetch the Service Config

- URL: https://api.powerbi.com/powerbi/globalservice/v201606/environments/discover?user=jeff@contoso.com
- This requires a Power BI email address, but does not require authentication. This includes the Authority URL, the Resource URL, Application ID, and Resource URI. These may be specific to your overall region, but should be fairly generic and aren't unique to your particular tenant.
- Typical values are:
  - Authority URL: https://login.windows.net/common/oauth2/authorize
  - Resource URL: https://analysis.windows.net/powerbi/api
  - Application ID: ea0616ba-638b-4df5-95b9-636659ae5121
  - Redirect URL: urn:ietf:wg:oauth:2.0:oob

Authenticate to Azure Active Directory using the above values and retrieve an access token.

Get your region's Service Backend URI after authenticating to the application using your access token.

- URL: https://api.powerbi.com/spglobalservice/GetOrInsertClusterUrisByTenantLocation
- Typical value: https://wabi-west-us-redirect.analysis.windows.net

Using the above access token and the region's Service Backend URI, query the Power Platform API using the GatewyClusters command to get all available information for your Power BI Gateway Clusters.

Typical URL for command: https://wabi-west-us-redirect.analysis.windows.net/unifiedgateway/gatewayclusters

After successfully performing all of the above steps, you will receive a JSON response from the service which contains all of the available information for the Gateway Clusters, including the names of the machines which have agents installed and what versions of the agent software they are running.

Sample JSON response from the gatewayclusters command:


```
[
  {
    "gatewayId":1234567,
    "objectId":"11111111-2222-3333-4444-555555555555",
    "name":"Test_DataGateway",
    "description":"Test enterprise data gateway",
    "publicKey":"asa894faj9fjawf8j4waf98jfa8ejpa9w8jf98sjf9s8jfa9e8fja9wfj8awf==",
    "keyword":null,
    "metadata":null,
```

"permission":{
"objectId":"aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaaaaaa",
"principalType":"User",
"tenantId":"bbbbbbbb-bbbb-bbbb-bbbb-bbbbbbbbbbbb",
"role":"Admin",
"allowedDataSourceTypes":[

]
},
"gateways":[
{
"gatewayId":1234567,
"gatewayObjectId":"11111111-2222-3333-4444-555555555555",
"gatewayName":"Test_DataGateway",
"gatewayAnnotation":"{\"gatewayContactInformation\":
[\"jeff@contoso.com\"],\"gatewayVersion\":\"3000.0.265\",\"gatewayWitnessString\":\"
{\\\"EncryptedResult\\\":\\\"AASDFAWAFsfadasfawfesfdsfs==\\\",\\\"IV\\\":\\\"aa3fasfeafasfafsa==\\\",\\\"Sigr

"gatewayStatus":"Installed",
"isAnchorGateway":true,
"gatewayClusterStatus":"Enabled",
"gatewayLoadBalancingSettings":null,
"gatewayStaticCapabilities":4064,
"gatewayPublicKey":"PASFUFHW7A398FAFJE9FS8DFJWA98FJFasdfalskfjaw8jfdjflk==",
"gatewayVersion":"3000.0.265",
"gatewayVersionStatus":"UpdateAvailable",
"expiryDate":null
},
{
"gatewayId":4444444,
"gatewayObjectId":"55555555-6666-7777-8888-999999999999",
"gatewayName":"Gateway 02",
"gatewayAnnotation":"{\"gatewayContactInformation\":
[\"jeff@contoso.com\"],\"gatewayVersion\":\"3000.9.194+ga90bb05c0e\",\"gatewayWitnessString\":\"
{\\\"EncryptedResult\\\":\\\"AASDFAWAFsfadasfawfesfdsfs==\\\",\\\"IV\\\":\\\"aa3fasfeafasfafsa\\\",\\\"Signatu

"gatewayStatus":"Installed",
"isAnchorGateway":false,
"gatewayClusterStatus":"Enabled",
"gatewayLoadBalancingSettings":null,
"gatewayStaticCapabilities":32736,
"gatewayPublicKey":"PASFUFHW7A398FAFJE9FS8DFJWA98FJFasdfalskfjaw8jfdjflk==",
"gatewayVersion":"3000.9.194",
"gatewayVersionStatus":"Latest",
"expiryDate":null
}
],
"loadBalancingSettings":null,
"annotation":"{\"gatewayContactInformation\":
[\"jeff@contoso.com\"],\"gatewayVersion\":\"3000.0.265\",\"gatewayWitnessString\":\"
{\\\"EncryptedResult\\\":\\\"Qasdofijw89jwaf8jslfkjaosfijawo==\\\",\\\"IV\\\":\\\"afasadfasfdfafd==\\\",\\\"Signa

```
    "versionStatus":"UpdateAvailable",
    "expiryDate":null,
    "type":null,
    "options":null
  }
]
```


Sample Application

To tie all of these steps together, I put together a brief C# console application which is capable of using stored credentials or prompting for credentials interactively, then performs the 4 steps listed above. Finally, the application receives the JSON response and de-serializes it and outputs it in a friendly way to the screen (or optionally inserts it into a SQL database table.

For the basics on how I perform the authentication and API query in the application, see my blog post here.

When launching the application and providing valid authentication (for a user with admin access to the Power BI Gateway, you should see output similar to the following, indicating that you successfully authenticated to Azure Active Directory and queried the Power Platform API for Power BI Gateway information:

```
- Retrieving data from: https://wabi-west-us-redirect.analysis.windows.net/unifiedgateway/gatewayclusters
   - Response code received: OK
   - De-serializing Gateway Data...

----------------------------------------------------------------
Gateway Cluster:
   Gateway Cluster Name: Test_GatewayCluster
   Gateway Cluster ID: 11111111-1111-1111-1111-111111111111
   Gateway Cluster Contact Email: jeff@jpries.com
   Gateway Cluster Department: IT
   Gateway Cluster Primary Machine: SERVER01
   Gateway Cluster Original Version: 3000.0.265

   Installed Agents:
      Agent #1 Name: Test_GatewayCluster
      Agent ID: 11111111-1111-1111-1111-111111111111
      Machine Name: SERVER01
      Department: IT
      Contact: jeff@jpries.com
      Agent Version: 3000.9.194
      Version Status: Latest
      Agent Status: Installed
      Clustering Status: Enabled
      Anchor (Primary) Gateway: True

      ---------------------------
      Agent #2 Name: Gateway 02 (SERVER02)
      Agent ID: 22222222-2222-2222-2222-222222222222
      Machine Name: SERVER02
      Department:
      Contact: jeff@jpries.com
      Agent Version: 3000.9.194
      Version Status: Latest
      Agent Status: Installed
      Clustering Status: Enabled
      Anchor (Primary) Gateway: False

----------------------------------------------------------------
```

Resources:

https://craigporteous.com/data-gateway-clustering/

https://docs.microsoft.com/en-us/data-integration/gateway/service-gateway-powershell-support

Posts in Series