**Quick Start Guide: Configuring Portal Usage Metrics Logging and Dashboard**

**Overview**

This guide outlines how to set up and configure portal usage metrics, including logging, feature layer creation, dashboard setup, and command-line configuration.

First of all, Go to **\\s1cfafile11.appservices.state.sbu\DataVizTeam\GeoState\Scripts\Tools\2. Portal Usage Metrics\Esri11UsageMetricsApp\EnterpriseMetricsNET\downloads**

All of the .net downloads are in the downloads folder of the EnterpriseMetricsNET package folder.

**Prerequisites:**

**Install .NET SDK**  
Verify you have the .NET SDK installed by running:

dotnet --version

If it’s not installed, download it from: <https://dotnet.microsoft.com/download>  
Install the latest Long-Term Support (LTS) version (e.g., .NET 6 or .NET 8).

**SDK downloaded sample**

PS C:\Windows\system32> dotnet --info

.NET SDK:

Version: 9.0.101

Commit: eedb237549

Workload version: 9.0.100-manifests.3068a692

MSBuild version: 17.12.12+1cce77968

Runtime Environment:

OS Name: Windows

OS Version: 10.0.19045

OS Platform: Windows

RID: win-x64

Base Path: C:\Program Files\dotnet\sdk\9.0.101\

.NET workloads installed:

There are no installed workloads to display.

Configured to use loose manifests when installing new manifests.

Host:

Version: 9.0.0

Architecture: x64

Commit: 9d5a6a9aa4

.NET SDKs installed:

3.1.426 [C:\Program Files\dotnet\sdk]

9.0.101 [C:\Program Files\dotnet\sdk]

.NET runtimes installed:

Microsoft.AspNetCore.App 3.1.32 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]

Microsoft.AspNetCore.App 9.0.0 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]

Microsoft.NETCore.App 3.1.32 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]

Microsoft.NETCore.App 8.0.8 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]

Microsoft.NETCore.App 9.0.0 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]

Microsoft.WindowsDesktop.App 3.1.32 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]

Microsoft.WindowsDesktop.App 9.0.0 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]

Other architectures found:

None

Environment variables:

Not set

global.json file:

Not found

Learn more:

https://aka.ms/dotnet/info

Download .NET:

<https://aka.ms/dotnet/download>

**Install the .NET Framework 4.7.2 Developer Pack**

1. **Download the Developer Pack**: Go to the official .NET Framework Developer Packs page:  
   [Download .NET Framework 4.7.2 Developer Pack](https://dotnet.microsoft.com/en-us/download/dotnet-framework/net472).
2. **Install the Developer Pack**:
   * Run the installer.
   * Follow the prompts to complete the installation.
3. **Verify Installation**: Open a new **Command Prompt** or **PowerShell** window and run:

dotnet --list-sdks

You should now have support for .NET Framework 4.7.2.

**Install .Net 6.0**

1. **Download .NET 6.0 Runtime**:  
   Go to the official .NET download page:  
   [Download .NET 6.0 Runtime (LTS)](https://dotnet.microsoft.com/download/dotnet/6.0).
2. **Choose the Installer**:  
   Select the **x64** version of the **.NET 6.0 Desktop Runtime** or **.NET Runtime** (whichever applies to your application).
3. **Install the Runtime**:  
   Run the installer and follow the prompts.

Verify Installation

Once installed, check if .NET 6.0 appears in the list of runtimes:

dotnet --list-runtimes

You should see:

Microsoft.NETCore.App 6.0.x [C:\Program Files\dotnet\shared\Microsoft.NETCore.

**1. Logging Setup**

**IIS Web Server Logging**

* **Prerequisites**: IIS logging must be enabled. Logs should use the **W3C format**.
* **Log Fields**:

#Fields: date time s-ip cs-method cs-uri-stem cs-uri-query s-port cs-username c-ip cs(User-Agent) cs(Referer) sc-status sc-substatus sc-win32-status time-taken

* **Default Log Directory**: Logs are stored in W3SVC# folders, where # is the IIS site ID. Example:

%SystemDrive%\inetpub\logs\LogFiles\W3SVC1

Graphical user interface, text, application

Description automatically generated

Log files will typically be placed in a folder called W3SVC# where the “#” is the number of the IIS site number where the portal web adaptor is hosted. For example, the “Default Web Site” has an ID of “1” so the log directory to be used would be: %SystemDrive%\inetpub\logs\LogFiles\W3SVC1.

Graphical user interface, text, application

Description automatically generated

**2. Feature Layer Configuration**

**Steps:**

1. Open **Portal Admin** and create a hosted feature layer.
   * **Template**: Select "File Geodatabase" from the "Database" deployment folder.
   * Publish the layer and fill in metadata (e.g., title, summary, tags).
   * **Note the itemId** for later configuration. Also make sure the table (featue layer) is shared publicly so the .net application can write to it.
2. **Optional**: Use an Enterprise Geodatabase with feature access and editing enabled.
   * Read-only layers (e.g., Map Image Layers) can also be configured for dashboard use.

Graphical user interface, text, application, email

Description automatically generated

\*\* Feature Layers may also be published from an Enterprise Geodatabase with feature access and editing enabled. The account defined as the “portalLogManager” must be able to edit the feature layer. A Map Image Feature Layer can be used for read only purposes, shared and configured with the Dashboard for broader consumption.

1. **Dashboard Setup**

**Steps:**

1. **Create a Dashboard**:
   * Use the provided template or create a blank dashboard in Portal for ArcGIS.
   * Copy the JSON configuration using a tool like AGO Assistant.
2. **JSON Updates**:
   * Replace placeholders in the template:
     + metrics-item-featurelayer: Replace with the itemId of the feature layer.
     + metrics--portal-url: Replace with your portal URL (e.g., https://mycompany.com/portal).
     + metrics-webmap-id: Replace with the web map ID (if applicable).
     + eb4634: Replace with your corporate brand color hex code.
3. Customize the dashboard further as needed.
4. **Portal Usage Metric Configuration**

Configure the command line interface using the portalLogConfig.json. You only need to fill out the logmanager and logger to get the tool running.

* portalLogManager – The portal log manager is the Portal for ArcGIS where metrics will be written to. This is the portal where the Hosted feature layer was created for storing the metrics. Complete the following:
  + url – The portal url, ie. <https://mydns.com/portal>
  + username – The portal admin username
  + password – The portal admin password
  + passwordEncrypted – This should be false by default. When false, the password should be the raw unencrypted portal admin password. On first run, the password will be encrypted, and this value will be set to true.
  + featureServiceItemId – This is the portal item id for the hosted feature layer that was created.
  + featureServiceLayerIndex – This is typically 0 and can be left as the default but if a custom hosted feature layer was created with multiple layer this index specified the numerical index for the layer within the service for where to write the metrics to.
* portalLogger– The portal logger is the Portal for ArcGIS for the portal where metrics will be captured. In a small organization the portalLogManager and portalLogger are typically the same environment. In a larger organization where multiple portals may exist, these can be different portal environments. Complete the following:
  + url – The portal url, ie. <https://mydns.com/portal>
  + username – The portal admin username
  + password – The portal admin password
  + passwordEncrypted – This should be false by default. When false, the password should be the raw unencrypted portal admin password. On first run, the password will be encrypted, and this value will be set to true.
  + name – This is and identifier for the portal where the metrics are being captured. This can be any text value such as “Prod” or any other identifier unique to this portal. Each metric captured will have this value set for filtering purposes in the dashboard.
  + logDirectory – This will be the directory where logs are written to by the portals Tomcat or IIS.
  + Tomcat -- This directory will be the same directory as defined in the server.xml set in the [Tomcat Logging](#_Tomcat_Logging).
  + IIS – This directory will be the logging directory used for the Portal Web Adaptor.

\*\*Note, because the config file is JSON format used path escape characters for paths. For example:

C:\\inetpub\\logs\\LogFiles\\W3SVC1

Or

C:/inetpub/logs/LogFiles/W3SVC1

* + logType – [Tomcat | IIS] This should define the log type being utilized to capture the metrics.
* userGroupMap – The user group map allows to use a set of regular expression on an ArcGIS Enterprise attribute such as username, idpUsername and email. It allows the use of a regular expressions to match values into a common group attribute to be used in the dashboard.
  + enabled –[true|false] to apply this logic to the metrics parsing operation.
  + ageSearchProperty – [usename | idpUsername | email] This is the value from the users portal profile to use to look up in the userGroupMap.
  + userGroupField – This is the field where the user group name will get stored in the hosted feature layer.
  + userGroupMap – This mapping allows for the definition of how to map a search attribute into group name. Regular Expressions can be used in the userGroupMap. For example,

“(?!)@esri.com$”: “Environmental Systems Research Institute”

This will map any user with a @esri.com email into “Environmental Systems Research Institute” group.

* + otherGroupValue – This allows the definition of a string to be applied for any users that do not meet a condition of the userGroupMap.
* filters – Various filters can be set for which item views to be ignored or not captured. This is intended to reduce noise in the dashboard of views that may not be wanted to be included in the metrics. Most should be left as the default but a few you may want to adjust:
  + viewsbyowner – true|false if true, any views to an item that are made by the items creator are ignored. For example, UserA creates a dashboard and is often looking at the dashboard adjusting and enhancements. Setting to true will not count UserA’s views of the dashboard.
  + viewsbyanonymous – true|false if true, any views to an item that have been made by an anonymous user will not be captured.
  + username – username is a list of usernames to ignore item views. This could be users of a core GIS team or other users whom you don’t want to track usage.
  + usergroupnames – usergroupnames is a list of portal group titles that you want to exclude from the metrics. For example, there is a portal group for an advanced set of users you would like to exclude from the dashboard. Regular Expressions can be used in this filter.
  + usergroupids – usergroupids is a list of portal group item ids that you want to exclude from the metrics. For example, there is a portal group for an advanced set of users you would like to exclude from the dashboard.
  + Itemowner – This is a list if item owners that can be used to exclude from metric collections. For example, the default configuration excludes any items owned by the esri user accounts. Regular Expressions can be used in this filter.
  + Itemid – This is a list of item ids of portal content that should be excluded from the metric collections.
  + Itemtype - This is a list of item types of portal content that should be excluded from the metric collections. For example, if you wanted to exclude Notebooks, you can use this to exclude all Notebook item types from the collection. See the “Type” column in this document <https://developers.arcgis.com/rest/users-groups-and-items/items-and-item-types/> for a list of item types. Regular Expressions can be used in this filter.
  + Itemtitle - This is a list of item titles of portal content that should be excluded from the metric collections. Regular Expressions can be used in this filter.
  + httpstatus – This is of http status codes that can be used to filter out unwanted metrics collections. Ranges of 400 and 500 are excluded by default.
* ldapConfg – if opting in to include LDAP or Active Directory as additional attributes to item views. Note, this requires the user of enterprise logins and uses the enterprise login for the viewer of the portal item to query additional attributes such as title, department or office that may add value to who is viewing a portal item.
  + enabled – true | false, this should be set to true if ldap look ups should be performed. Leave as false to not use this functionality.
  + server – This will be the LDAP server to query for information.
  + username – The LDAP username. This often is a service account specifically used to query LDAP.
  + password – The LDAP password
  + passwordEncrypted – This should be false by default. When false, the password should be the raw unencrypted LDAP account password. On first run, the password will be encrypted, and this value will be set to true.
  + domain – The LDAP user account domain.
  + searchProperty – This is the search property that is used to query LDAP. This is often userprincipalname or samaccountname.
  + ageSearchProperty – [ usename | idpUsername | email ] This is the value from the users portal profile to use to look up in the LDAP searchProperty.
  + searchRemoveDomain – true|false, this tells the LDAP config to remove the users domain from the username before querying LDAP. For example, the ArcGIS Enterprise login is [jus11031@esri.com](mailto:jus11031@esri.com). If true, the domain will be removed and only jus11031 will be used to query the “searchProperty” specified above to locate the user.
  + searchDistinquishedNames – This is a list of one or more distinguished names to locate a user in active directory. For example, CN=Users,DC=domain,DC=net will be the directory searched to locate the user.
  + attributeMap – This is a dictionary of key value pairs for fields to pull from ldap (the key) and the field attribute name for where to store the value in the hosted feature layer (the value). For example, the following will get the “title” attribute value from LDAP and store it in a field in called “user\_title” field in the hosted feature layer. NOTE: This field must be added to the hosted feature layer in the Portal user interface.

{ “title”: “user\_title”}

* locationMap – This is a look up to match a coordinate value to a specific LDAP attribute to show spatial locations on a map. For example, an office city value is pulled from LDAP for the user, a list of coordinates for office locations can be provided to add a geometry to the portal item view so they can be represented in a map.
  + enabled – true|false, true to enable this feature, false to ignore it.
  + locationAttribute – the LDAP attribute to retrieve for the user. For example, City.
  + locationWKID – The coordinates well-known id for the coordinate system used in the locations lookup.
  + locations – A list of locations where the “name” is the value that will be mached for the locationAttribute specified above. The “x” and “y” values are the coordinate to be used for that given location in the coordinates of the well-known id.
* locationGeocode -- This is a similar process to the locationMap except the LDAP attributes specified will be used to geocode the location using the portals defined geocoder.
  + enabled – true|false, true to enable this feature, false to ignore it.
  + streetAddressAttribute – (optional) the layer field in the attributeMap if configured that contain a street address.
  + cityAttribute – (optional) the layer field in the attributeMap if configured that contain a city.
  + stateAttribute – (optional) the layer field in the attributeMap if configured that contain a state.
  + zipAttribute – (optional) the layer field in the attributeMap if configured that contain a postal code.
  + countryAttribute – (optional) the layer field in the attributeMap if configured that contain a country.
  + geocodeProvider – (optional) the geocodeProvider is an optional object field. It contains advanced properties to configure the geocoder to use. If not provided, it defaults to using the geocoder defined as the default with the ArcGIS Enterprise configured with the portalLogManager.

\*\* This will require credits for AGOL or the use of a created or owned geocoder such as Street Map Premium.

* + - * geocodeProviderType – [PortalLogManager | PortalLogger | GeocodePortal | APIKey] – The type of value set defines what geocoder to use:
        + “PortalLogManager” – This will use the credentials defined in the portalLogManager default geocoder. This is the default.
        + “PortalLogger” – This will use the credentials defined in portalLogger default geocoder.
        + “GeocodePortal” – This will allow you to define a custom connection to use for the geocoder, like an ArcGIS Online Organization.
        + “APIKey” – This will allow you to define a API Key generated through ArcGIS Online. If using APIKEY, you must define the “geocodeURL” property for the geocode service to use AND “geocodeAPIKey” values.
      * geocodeURL – The geocode url to use if not using the default from PortalLogManager, PortalLogger, or GeocodePortal. For example, if you want to use Street Map Premium as a geocode service that is not configured as the DEFAULT geocoder with the Portal for ArcGIS. This could also be the ArcGIS Online World Geocoder and used in conjunction with the API Key.
      * geocodeAPIKey – The API Key to use as the “token” parameter for geocoding.
      * geocodeAPIKeyReferer – The API Key referer header to use with the API Key if a referrer has been defined when generating the API Key.
      * apiKeyEncrypted – [ true | false ] This should be false by default. When false, the api key should be the raw unencrypted api key. On first run, the api key will be encrypted, and this value will be set to true.

The following are some examples:

PortalLogger – Using default geocode url

"geocodeProvider": {

"geocodeProviderType": "PortalLogger",

},

API Key

"geocodeProvider": {

"geocodeProviderType": "APIKey",

"geocodeURL": "https://geocode.arcgis.com/arcgis/rest/services/World/GeocodeServer",

"geocodeAPIKey": "myencryptedkey”

"apiKeyEncrypted": true

},

GeocodePortal

"geocodeProvider": {

"geocodeProviderType": "GeocodePortal",

"geocodePortal": {

"url": "https://myagolorg.maps.arcgis.com",

"username": "myAGOLusername",

"password": "myencryptedpassword

"passwordEncrypted": true

},

"geocodeURL": "https://geocode.arcgis.com/arcgis/rest/services/World/GeocodeServer",

},

* + - * nullGeoX – (Optional) The x coordinate value to use as “null” location when a geocode is not found. – Defaults to 0 if not set.
      * nullGeoY – (Optional) The y coordinate value to use as “null” location when a geocode is not found. – Defaults to 0 if not set.
      * nullGeoWKID – (Optional) The WKID of the coordinates used for “null” value. – Defaults to 4326 (WGS84) if not set.
      * nullCache – [true | false] Geocode locations are cached so they can be reused for users in the same location. If a location is not found, it can cache that location so it’s not attempting to geocode again (ie. potentially consuming credits for search locations that have already not been found). The default is true to not search again for a location not found. Set to false if it is desired for the geocode to run again.
    - azzureADConfig - if opting in to include Azure Active Directory as additional attributes to item views. Note, this requires the use of enterprise logins and uses the enterprise login for the viewer of the portal item to query additional attributes such as title, department or office that may add value to who is viewing a portal item.
  + enabled – true | false, this should be set to true if ldap look ups should be performed. Leave as false to not use this functionality.
  + clientId – This will be the LDAP server to query for information.
  + clientSecret – The LDAP username. This often is a service account specifically used to query LDAP.
  + clientSecretEncrypted – This should be false by default. When false, the password should be the raw unencrypted LDAP account password. On first run, the password will be encrypted, and this value will be set to true.
  + tenantId– The LDAP user account domain.
  + authorityHost– This is the search property that is used to query LDAP. This is often userprincipalname or samaccountname.
  + ageSearchProperty – [ usename | idpUsername | email ] This is the value from the users portal profile to use to look up in the LDAP searchProperty.
  + attributeMap – This is a dictionary of key value pairs for fields to pull from azure ad (the key) and the field attribute name for where to store the value in the hosted feature layer (the value). For example, the following will get the “title” attribute value from Azure AD and store it in a field in called “user\_title” field in the hosted feature layer. NOTE: This field must be added to the hosted feature layer in the Portal user interface.

{ “title”: “user\_title”}

* locationMap – This is a look up to match a coordinate value to a specific Azure AD attribute to show spatial locations on a map. For example, an office city value is pulled from Azure AD for the user, a list of coordinates for office locations can be provided to add a geometry to the portal item view so they can be represented in a map. – See “locationMap” in LDAP section above for more information.
  + locationGeocode -- This is a similar process to the locationMap except the Azure AD attributes specified will be used to geocode the location using the portals defined geocoder. . – See “locationGeocode” in LDAP section above for more information.
    - tagConfig – This is a way to grab tags from the portal item being viewed and associated it with the portal item view. For example, tags are added to portal items like “FY:2004”. This tag if found on the portal item will split on the delimiter, in this case the “:” and store the value 2004 in a hosted feature layer field called “fy”. NOTE: Field defined must be added to the hosted feature layer in the Portal user interface.
      * delimiter – the delimiter character used to split the tag.
      * tags – a list of tag names to capture with a portal item is viewed. In the example case the “fy” tag will be specified as a list value. This can be left as an empty list to ignore.

**5. Running the Solution**

**Navigate to the Solution Directory**

* Open **Command Prompt** or **PowerShell**.
* Use the cd command to navigate to the folder containing the **PortalUsageMetrics.sln** file:

cd path\to\PortalUsageMetrics

**Restore Dependencies**

Run the following command to restore all required dependencies and NuGet packages:

dotnet restore

Sample Success

PS C:\Users\DiazND\Documents\enterprise-metrics-2\PortalUsageMetrics> dotnet restore

Welcome to .NET 9.0!

---------------------

SDK Version: 9.0.101

Telemetry

---------

The .NET tools collect usage data in order to help us improve your experience. It is collected by Microsoft and shared with the community. You can opt-out of telemetry by setting the DOTNET\_CLI\_TELEMETRY\_OPTOUT environment variable to '1' or 'true' using your favorite shell.

Read more about .NET CLI Tools telemetry: https://aka.ms/dotnet-cli-telemetry

----------------

Installed an ASP.NET Core HTTPS development certificate.

To trust the certificate, run 'dotnet dev-certs https --trust'

Learn about HTTPS: https://aka.ms/dotnet-https

----------------

Write your first app: https://aka.ms/dotnet-hello-world

Find out what's new: https://aka.ms/dotnet-whats-new

Explore documentation: https://aka.ms/dotnet-docs

Report issues and find source on GitHub: https://github.com/dotnet/core

Use 'dotnet --help' to see available commands or visit: https://aka.ms/dotnet-cli

--------------------------------------------------------------------------------------

Restore complete (30.3s)

Build succeeded in 31.0s

**Build the Solution**

Build the solution in **Release mode** to produce optimized binaries:

dotnet build --configuration Release

* This will compile all the projects in the solution.
* Check for success messages. If there are errors, they will be displayed for troubleshooting. Typically, it will say, 1 error and 75 warnings. If it has an error, you will need to rectify the issue before moving on.

**6. Update RESTHandler.cs (Optional)**

This was done already, the OG.cs scripts lives in the archive folder. However the steps are listed below if you have a fresh version of the metrics tool. The default version does not have IWA enabled in the csharp script.

To **enable Integrated Windows Authentication (IWA)** for your RestHandler class and remove reliance on username/password, you need to configure the HttpClient to use the **default Windows credentials**.

Here are the steps to modify your RestHandler:

**Update HttpClientHandler to Use Default Credentials**

Add UseDefaultCredentials = true to the HttpClientHandler so that it automatically picks up the logged-in Windows user credentials (whoami).

**Updated clientHandler Configuration:**

private static readonly HttpClientHandler clientHandler = new()

{

UseDefaultCredentials = true, // Enables Integrated Windows Authentication (IWA)

PreAuthenticate = true, // Ensures credentials are sent upfront

ServerCertificateCustomValidationCallback = (sender, cert, chain, sslPolicyErrors) => { return true; }

};

**Remove Username/Password from JSON**

Since IWA does not need explicit credentials, ensure the JSON configuration removes the username/password fields for portalLogManager and portalLogger

**7. Run the Tool**

Run the **pumcli** project using the following command:

dotnet run --project pumcli\pumcli.csproj

* If the tool requires **arguments**, include them after --. Example:

dotnet run --project pumcli\pumcli.csproj -- --arg1 value1 --arg2 value2

* If it generates files or logs, check the **output folder** (default is \bin\Release or a location specified in the tool).

**How to Use the Tool**

The available operations are:

* InitConfig
* Validate
* Parse
* ParseTokens
* ServerXMLEntry
* PayloadSample
* LDAPSampleOutput
* LDAPEnrich
* AzureADSampleOutput
* RuntimeInfo

**Basic Command Syntax**

Here’s how to run the tool with an operation:

dotnet run --project pumcli\pumcli.csproj -- -o <operation> -c <config\_file\_path>

* Replace <operation> with one of the valid operations, e.g., Validate or Parse.
* Replace <config\_file\_path> with the path to the portal log configuration file.

**Examples**

**Run the Validate Operation**

If you have a configuration file named config.json, run:

dotnet run --project pumcli\pumcli.csproj -- -o Validate -c F:\enterprise-metrics-netapp\PortalUsageMetrics\portalconfig.json

**Run the Parse Operation**

dotnet run --project pumcli\pumcli.csproj -- -o Parse -c F:\enterprise-metrics-netapp\PortalUsageMetrics\portalconfig.json

**Additional Notes**

* Ensure the config file password isn’t encrypted on the initial run.
* Once you run the initial run on the config file, it will add more to the configuration file to fill out for ldap considerations.
* Had to add my account to the security permissions for inetpub folder (also click through each folder till you see the logs to active your permissions)
* If it is not shared, you need to create a network share:
* On s1cfagisprt03, share the C:\inetpub\logs folder:
* Right-click → Properties → Sharing tab → Advanced Sharing → Share the folder.
* Set permissions for your account to at least Read access.
* If the operation needs --username for LDAP operations, you can include it:

dotnet run --project pumcli\pumcli.csproj -- -o LDAPEnrich -u myUser -c C:\path\to\config.json

* Add the -v flag for verbose output. PS C:\Users\DiazND\Documents\enterprise-metrics-2\PortalUsageMetrics> dir \*.log -Recurse
* Directory: C:\Users\DiazND\Documents\enterprise-metrics-2\PortalUsageMetrics\pumcli\bin\Debug\net6.0\logs

**Output Verification**

* Check the terminal for outputs, logs, or status messages.
* If the tool generates metrics files, they will typically appear in:

path\to\PortalUsageMetrics\pumcli\bin\Release\net6.0\

Replace net6.0 with the correct .NET runtime version used in your project. Check your logs and gdb table on Portal!!!

**Troubleshooting Tips**

1. **Missing SDK**:  
   If you see *“No SDK found”* or similar errors, install the latest .NET SDK.
2. **Project Errors**:  
   If there’s an issue with the pumcli project, rebuild everything cleanly:

dotnet clean

dotnet restore

dotnet build --configuration Release

1. **Execution Errors**:  
   If the tool fails to run, ensure all required configurations, environment variables, or input files are correctly set.
2. **Dependencies**:  
   Ensure any third-party dependencies (like database connections or APIs) are reachable and configured.

**Summary of Commands**

Here are all the commands together for quick reference:

# Navigate to the solution folder

cd path\to\PortalUsageMetrics

# Restore dependencies

dotnet restore

# Build the solution

dotnet build --configuration Release

# Run the main project

dotnet run --project pumcli\pumcli.csproj