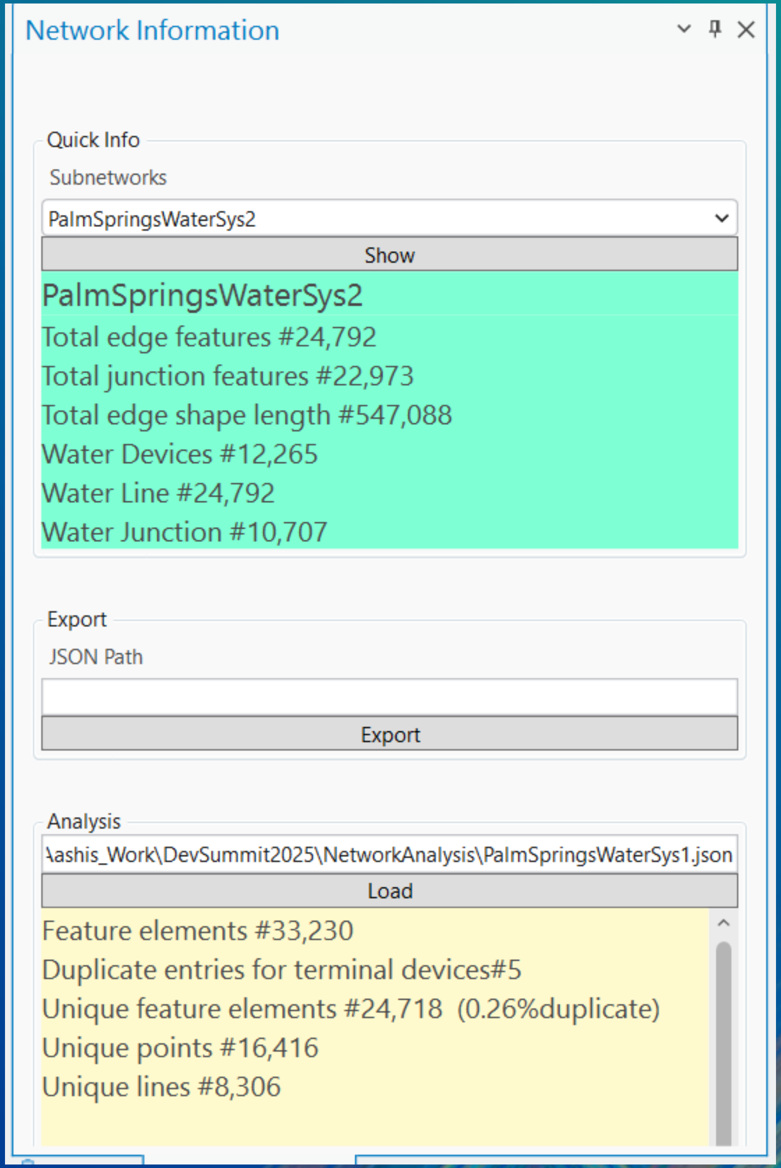
ArcGIS Pro SDK for .NET: Exporting Utility Networks for Network Analysis in the Pro SDK

Aashis Lamsal

2025 ESRI DEVELOPER & TECHNOLOGY SUMMIT

Agenda

- Background
- Extract Network Data
- Analyze Network Data
- Demo
- Questions



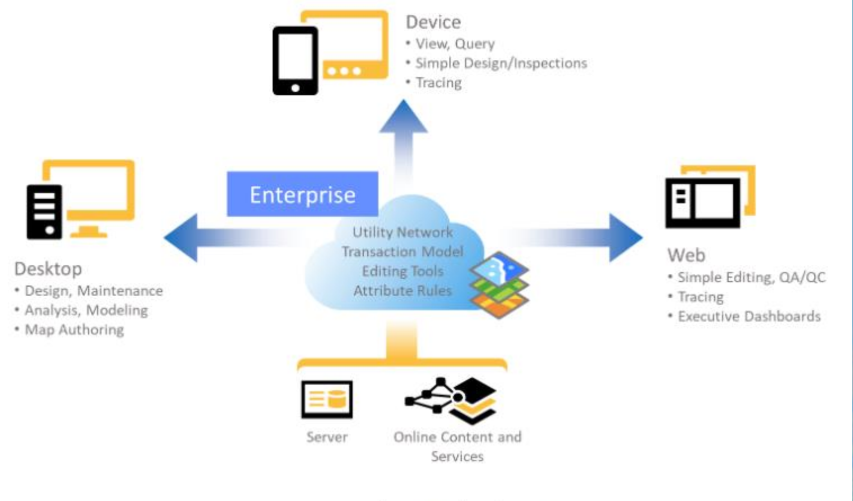
PalmSpringsWaterSys2
Total edge features #24,792
Total junction features #22,973
Total edge shape length #547,088
Water Devices #12,265
Water Line #24,792
Water Junction #10,707

\ashis_Work\DevSummit2025\NetworkAnalysis\PalmSpringsWaterSys1.json

Feature elements #33,230
Duplicate entries for terminal devices#5
Unique feature elements #24,718 (0.26%duplicate)
Unique points #16,416
Unique lines #8,306

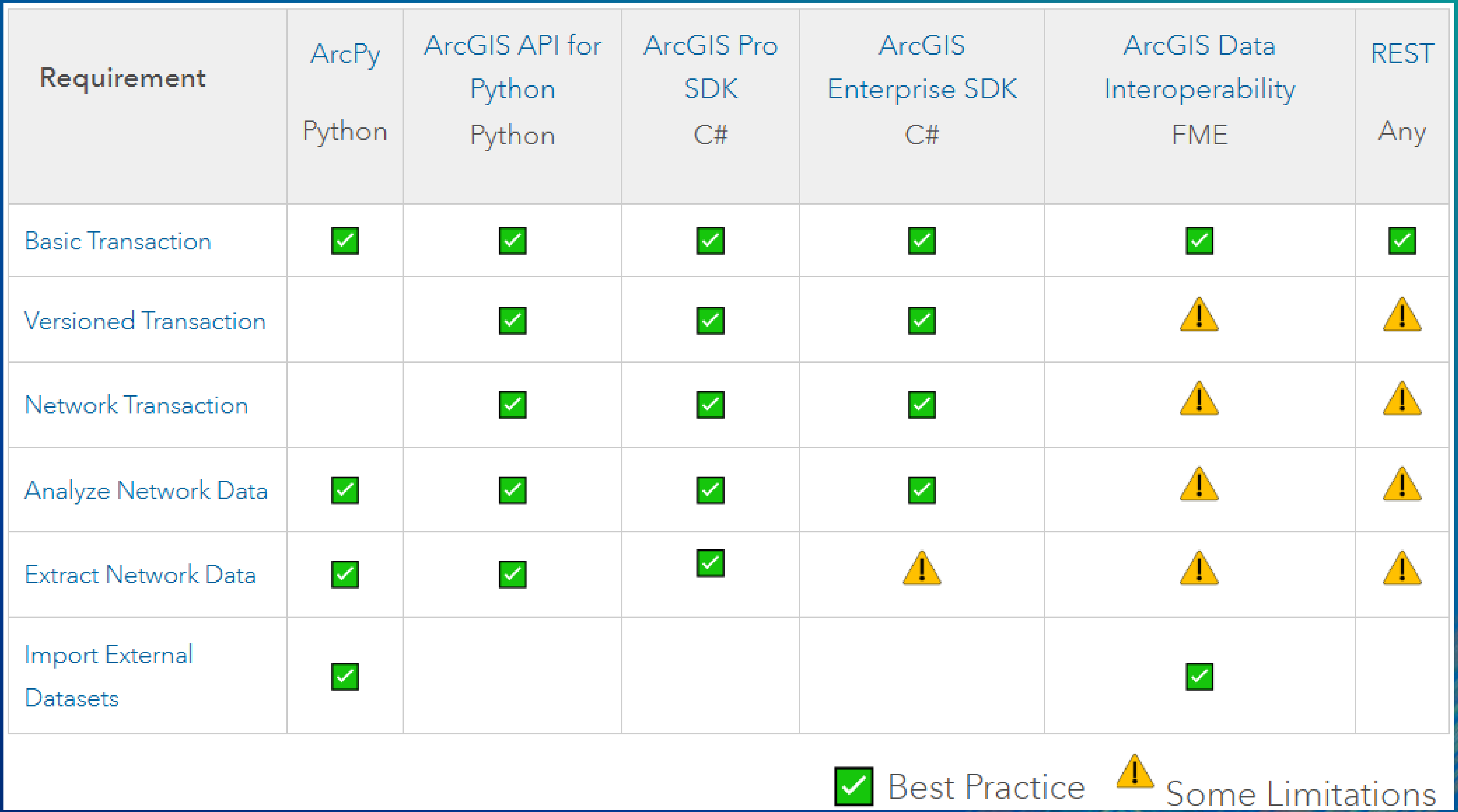
Background

- Enterprise GIS is a powerful tool for integration because it allows a company to model the logical, spatial, and topological relationships for all their data.
- Utility data is maintained in the GIS
  - System of record
  - Location and configuration of point and linear assets
  - Connectivity between assets and systems (OMS & DMS)
- Extract network information from a utility network for analysis
  - Specialized network models
  - Require engineering characteristics for each feature
  - Often require historical measurements (voltage, flow, etc)



Utility Network Architecture

UN APIs

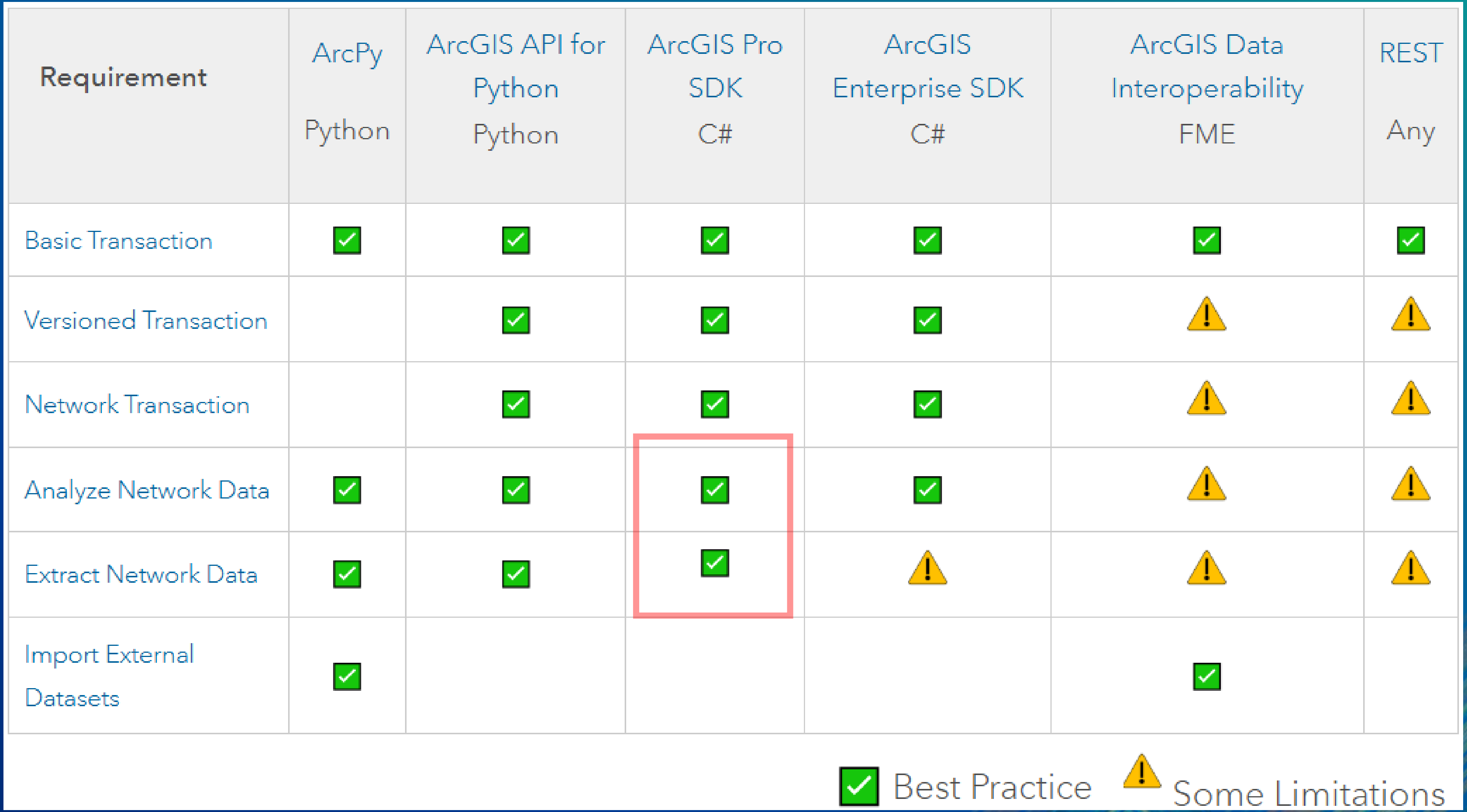


Requirement	ArcPy Python	ArcGIS API for Python Python	ArcGIS Pro SDK C#	ArcGIS Enterprise SDK C#	ArcGIS Data Interoperability FME	REST Any
Basic Transaction	✓	✓	✓	✓	✓	✓
Versioned Transaction		✓	✓	✓	⚠	⚠
Network Transaction		✓	✓	✓	⚠	⚠
Analyze Network Data	✓	✓	✓	✓	⚠	⚠
Extract Network Data	✓	✓	✓	⚠	⚠	⚠
Import External Datasets	✓				✓	

✓ Best Practice ⚠ Some Limitations

Blog:Journey to the Utility Network: Integrations Overview

UN APIs



Requirement	ArcPy Python	ArcGIS API for Python Python	ArcGIS Pro SDK C#	ArcGIS Enterprise SDK C#	ArcGIS Data Interoperability FME	REST Any
Basic Transaction	✓	✓	✓	✓	✓	✓
Versioned Transaction		✓	✓	✓	⚠	⚠
Network Transaction		✓	✓	✓	⚠	⚠
Analyze Network Data	✓	✓	✓	✓	⚠	⚠
Extract Network Data	✓	✓	✓	⚠	⚠	⚠
Import External Datasets	✓				✓	

✓ Best Practice ⚠ Some Limitations

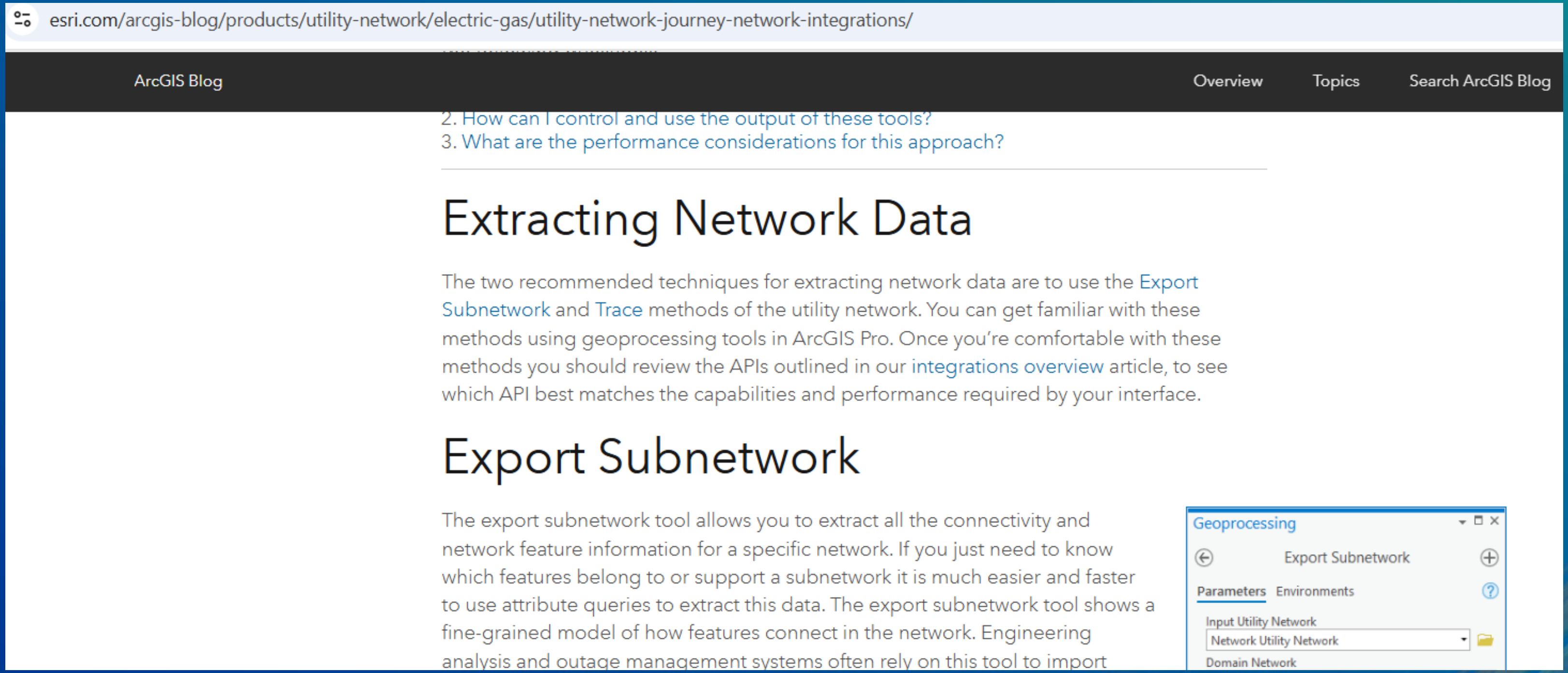
Blog:Journey to the Utility Network: Integrations Overview

Extract Network Data

- Network data can be extracted by trace export or export subnetwork
  - Trace (connected) – everything that is connected to the network
  - Trace (subnetwork) – subset of the network
  - Export subnetwork - subset of the network
- JSON
  - Include features, attribution, and connectivity
- Usage
  - Engineering analysis & planning
  - Integration with third-party systems for outage or distribution management
  - Connectivity analysis

Extract Network Data

- Export Trace
  - Dirty or clean subnetwork
  - Connected trace
- Export Subnetwork
  - Clean subnetwork



esri.com/arcgis-blog/products/utility-network/electric-gas/utility-network-journey-network-integrations/

2. How can I control and use the output of these tools?
3. What are the performance considerations for this approach?

Extracting Network Data

The two recommended techniques for extracting network data are to use the Export Subnetwork and Trace methods of the utility network. You can get familiar with these methods using geoprocessing tools in ArcGIS Pro. Once you're comfortable with these methods you should review the APIs outlined in our integrations overview article, to see which API best matches the capabilities and performance required by your interface.

Export Subnetwork

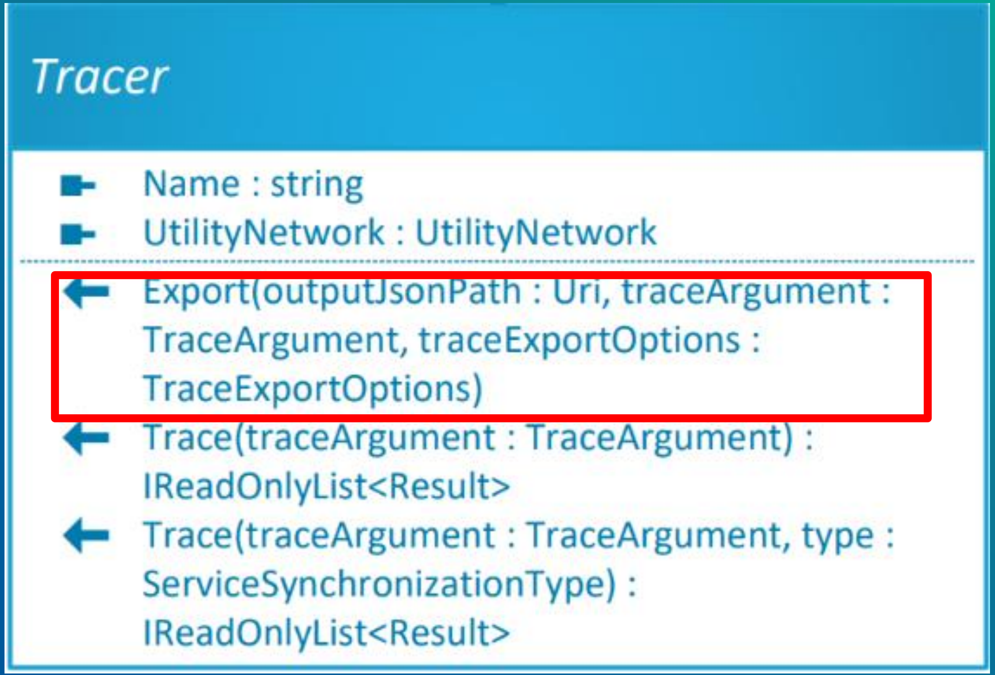
The export subnetwork tool allows you to extract all the connectivity and network feature information for a specific network. If you just need to know which features belong to or support a subnetwork it is much easier and faster to use attribute queries to extract this data. The export subnetwork tool shows a fine-grained model of how features connect in the network. Engineering analysis and outage management systems often rely on this tool to import

[Blog: Journey to the Utility Network: Integrations Overview]

Export Trace

Export Trace results as JSON

- Export Trace
  - Dirty or clean subnetwork
  - Connected trace



Tracer

Name : string
UtilityNetwork : UtilityNetwork

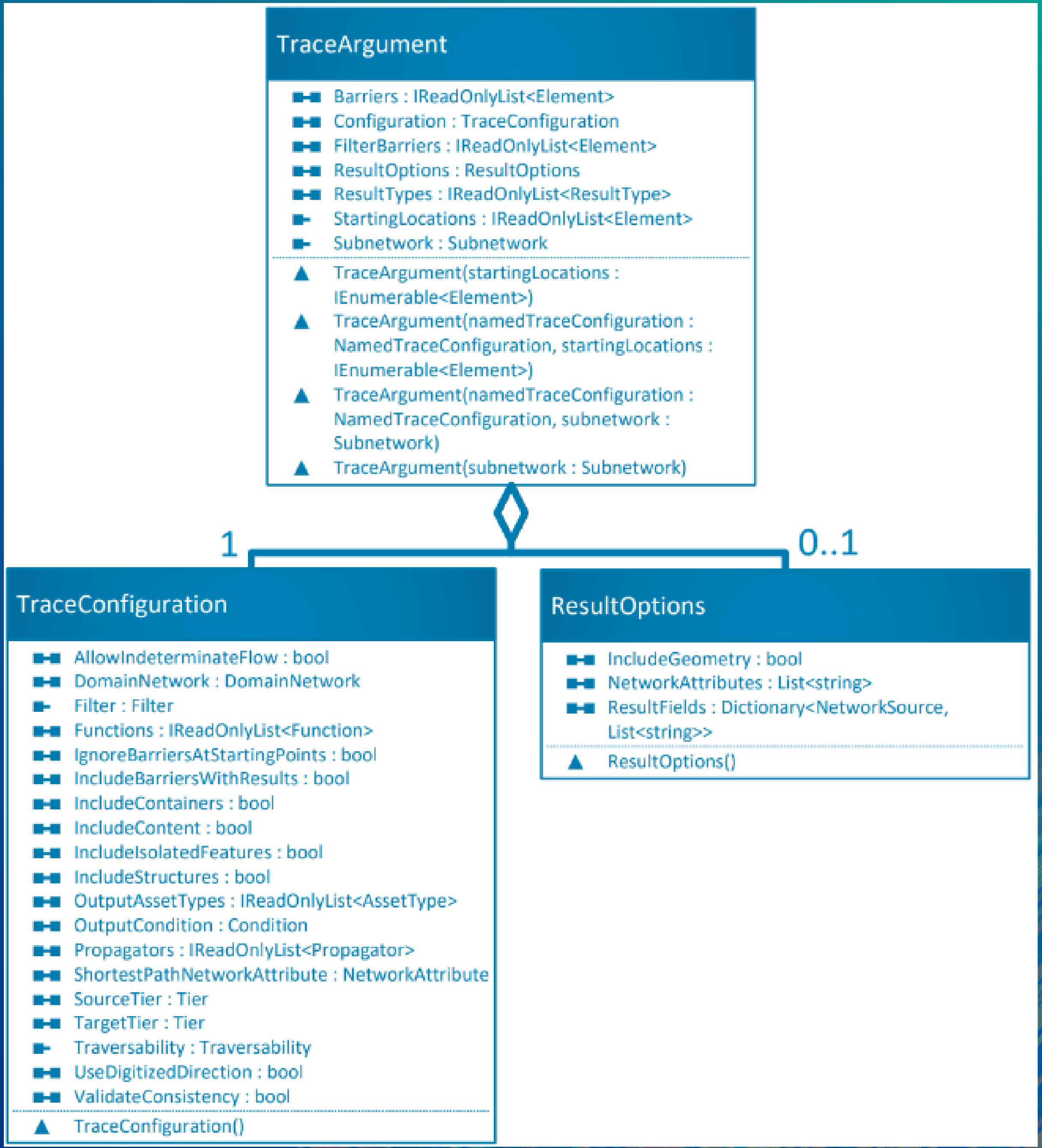
Export(outputJsonPath : Uri, traceArgument : TraceArgument, traceExportOptions : TraceExportOptions)
Trace(traceArgument : TraceArgument) : IReadOnlyList<Result>
Trace(traceArgument : TraceArgument, type : ServiceSynchronizationType) : IReadOnlyList<Result>

- `UtilityNetwork.GetTraceManager(): TraceManager`
- `TraceManager.GetTracer():Tracer`
- `Tracer.Export(Uri outputJsonPath, TraceArgument traceArgument, TraceExportOptions traceExportOptions)`

Export Trace

Export Trace results as JSON

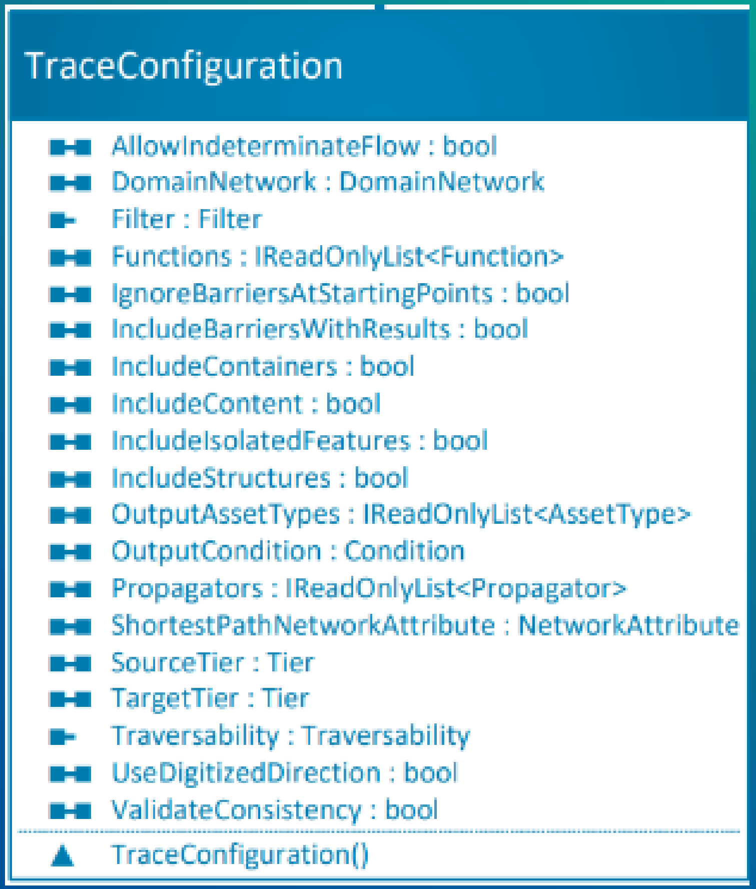
- Define trace argument options
  - Starting elements
  - Subnetwork
- `TraceArgument` class
  - `Subnetwork`
  - `ResultTypes`
  - `ResultOptions`
  - `Barriers`
  - `TraceConfiguration`
    - `UseDigitizedDirection`



```
TraceArgument
  Barriers : IReadOnlyList<Element>
  Configuration : TraceConfiguration
  FilterBarriers : IReadOnlyList<Element>
  ResultOptions : ResultOptions
  ResultTypes : IReadOnlyList<ResultType>
  StartingLocations : IReadOnlyList<Element>
  Subnetwork : Subnetwork
  ----
  TraceArgument(startingLocations : IEnumerable<Element>)
  TraceArgument(namedTraceConfiguration : NamedTraceConfiguration, startingLocations : IEnumerable<Element>)
  TraceArgument(namedTraceConfiguration : NamedTraceConfiguration, subnetwork : Subnetwork)
  TraceArgument(subnetwork : Subnetwork)

TraceConfiguration
  AllowIndeterminateFlow : bool
  DomainNetwork : DomainNetwork
  Filter : Filter
  Functions : IReadOnlyList<Function>
  IgnoreBarriersAtStartingPoints : bool
  IncludeBarriersWithResults : bool
  IncludeContainers : bool
  IncludeContent : bool
  IncludeIsolatedFeatures : bool
  IncludeStructures : bool
  OutputAssetTypes : IReadOnlyList<AssetType>
  OutputCondition : Condition
  Propagators : IReadOnlyList<Propagator>
  ShortestPathNetworkAttribute : NetworkAttribute
  SourceTier : Tier
  TargetTier : Tier
  Traversability : Traversability
  UseDigitizedDirection : bool
  ValidateConsistency : bool
  ----
  TraceConfiguration()

ResultOptions
  IncludeGeometry : bool
  NetworkAttributes : List<string>
  ResultFields : Dictionary<NetworkSource, List<string>>
  ----
  ResultOptions()
```

Export Trace

Export Trace results as JSON



TraceConfiguration

```
AllowIndeterminateFlow : bool
DomainNetwork : DomainNetwork
Filter : Filter
Functions : IReadOnlyList<Function>
IgnoreBarriersAtStartingPoints : bool
IncludeBarriersWithResults : bool
IncludeContainers : bool
IncludeContent : bool
IncludeIsolatedFeatures : bool
IncludeStructures : bool
OutputAssetTypes : IReadOnlyList<AssetType>
OutputCondition : Condition
Propagators : IReadOnlyList<Propagator>
ShortestPathNetworkAttribute : NetworkAttribute
SourceTier : Tier
TargetTier : Tier
Traversability : Traversability
UseDigitizedDirection : bool
ValidateConsistency : bool
TraceConfiguration()
```

- `TraceConfiguration` class
  - The domain network on which the trace will run
  - Traversability barriers
  - A filter for the type of assets included in the results
  - Functions to compute while performing the trace

- Subnetwork trace, a default trace configuration can be obtained from a utility network tier with the names of the domain network and the tier.

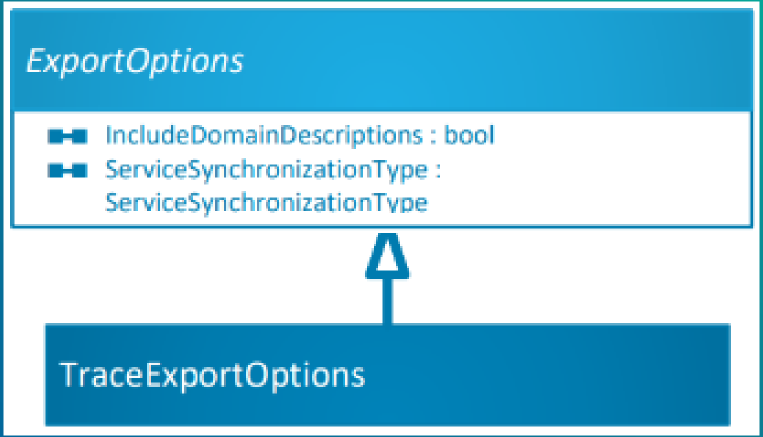
```
// Get trace configuration from Tier
TraceConfiguration traceConfiguration = tier.GetTraceConfiguration();
```

Export Trace

Export Trace results as JSON

- Define trace export options
- Set output path
- Save trace results as a JSON file

```
ServiceSynchronizationType
  SynchronousService
  AsynchronousService
```



ExportOptions
IncludeDomainDescriptions : bool
ServiceSynchronizationType : ServiceSynchronizationType
TraceExportOptions

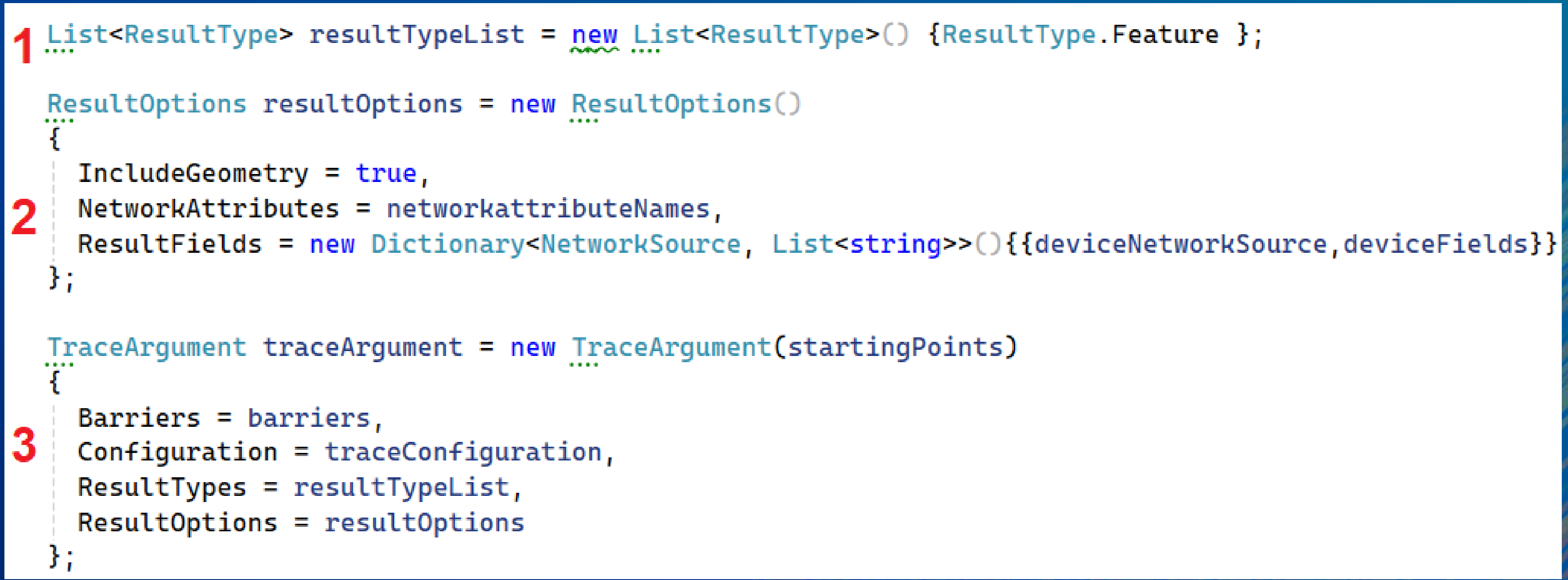
```
// Set export options
TraceExportOptions exportOptions = new TraceExportOptions()
{
  ServiceSynchronizationType = ServiceSynchronizationType.Asynchronous,
  IncludeDomainDescriptions = true,
};

// Path to save trace results
string jsonPath =  $"{Path.GetTempPath()}TraceResults.json";
Uri jsonUri = new Uri(jsonPath);

// Execute export
downstreamTracer.Export(jsonUri, traceArgument, exportOptions);

string jsonAbsolutePath = HttpUtility.UrlDecode(jsonUri.AbsolutePath);
if (jsonUri.IsFile && File.Exists(jsonAbsolutePath))
{
  // Work with the JSON
}
```

Feature-level Info From a Trace

- Fetch feature information during a trace
- Why?
  - Avoids the overhead of querying
    - Source feature class
    - Network attributes
- Existing workaround
  - Make multiple calls to the network source feature classes
  - Each of these is a round trip to the server

Feature-level Info From a Trace

- Three steps
  - Set ResultType
  - Set ResultOptions
    - NW Attributes & FeatureClass Fields
  - Set TraceArgument & call Trace()



```
1 List<ResultType> resultTypeList = new List<ResultType>() {ResultType.Feature };

  ResultOptions resultOptions = new ResultOptions()
  {
    IncludeGeometry = true,
2   NetworkAttributes = networkattributeNames,
    ResultFields = new Dictionary<NetworkSource, List<string>>(){{deviceNetworkSource,deviceFields}}
  };

  TraceArgument traceArgument = new TraceArgument(startingPoints)
  {
    Barriers = barriers,
3   Configuration = traceConfiguration,
    ResultTypes = resultTypeList,
    ResultOptions = resultOptions
  };
```

Demo

Export Subnetwork

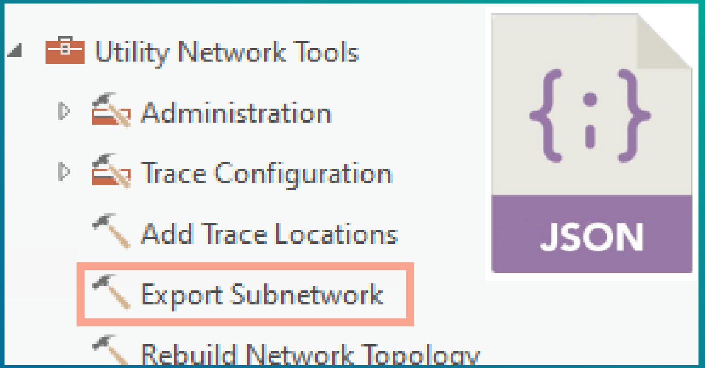
Subnetwork as JSON

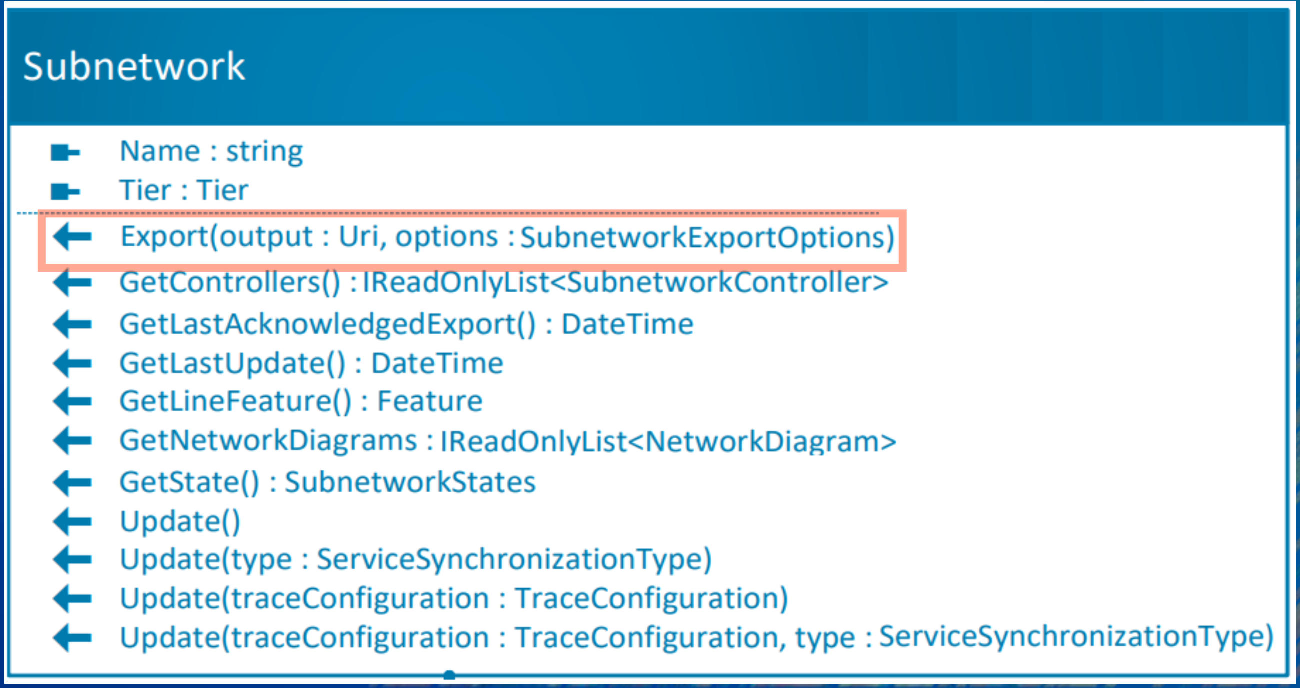
Export to a JSON file

- Subnetwork as a JSON file

```
Subnetwork.Export(Uri outputJsonPath, SubnetworkExportOptions options)
```

- Supplements the existing GP Tool





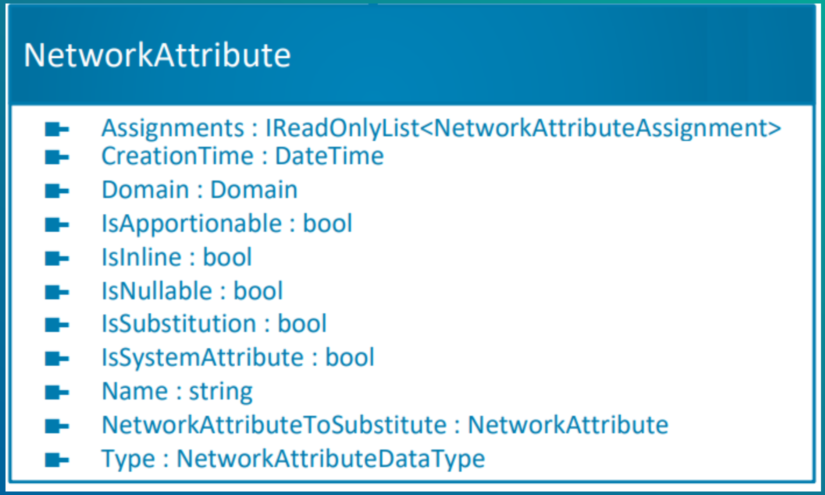
Subnetwork

Name : string
Tier : Tier
Export(output : Uri, options : SubnetworkExportOptions)
GetControllers() : IReadOnlyList<SubnetworkController>
GetLastAcknowledgedExport() : DateTime
GetLastUpdate() : DateTime
GetLineFeature() : Feature
GetNetworkDiagrams : IReadOnlyList<NetworkDiagram>
GetState() : SubnetworkStates
Update()
Update(type : ServiceSynchronizationType)
Update(traceConfiguration : TraceConfiguration)
Update(traceConfiguration : TraceConfiguration, type : ServiceSynchronizationType)

Subnetwork as JSON

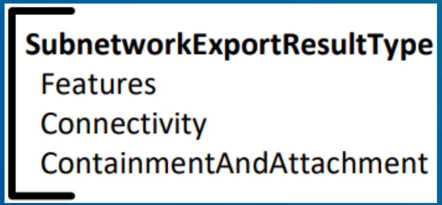
Export Options

- Defines the options to export a subnetwork
- `SubnetworkExportOptions` class
  - Domain descriptions
  - Shape
  - NW Attributes
  - NW source attribute fields
  - Result types

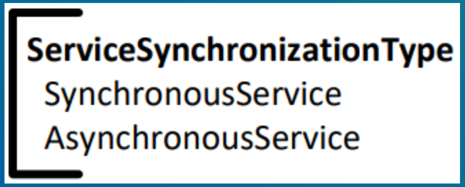


NetworkAttribute

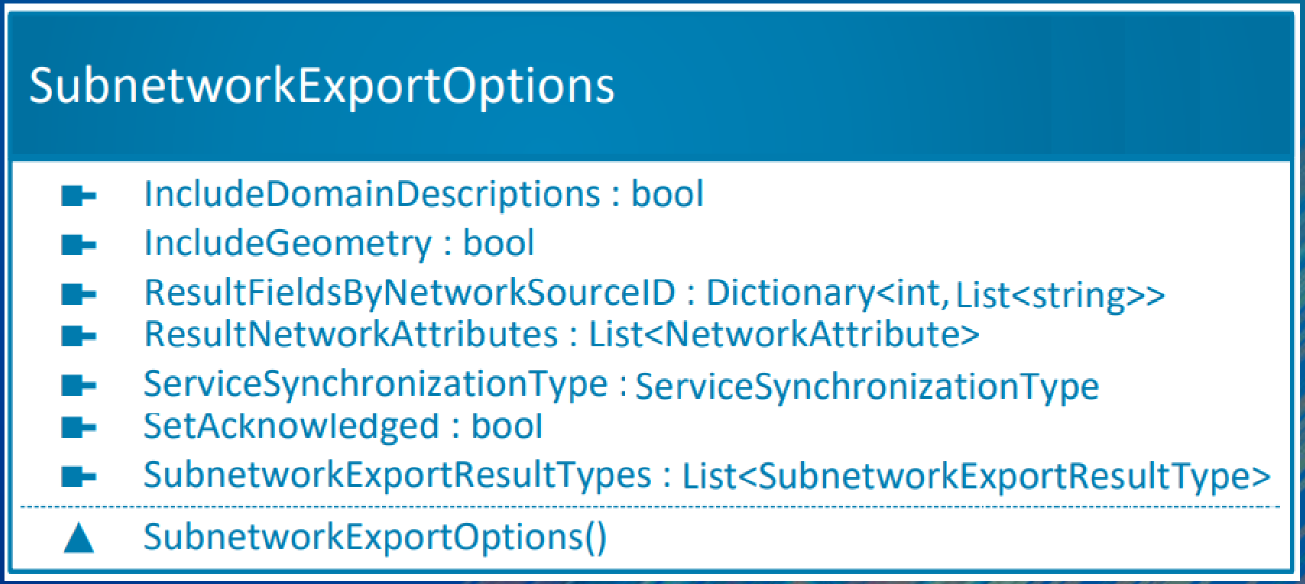
- Assignments : IReadOnlyList<NetworkAttributeAssignment>
- CreationTime : DateTime
- Domain : Domain
- IsApportionable : bool
- IsInline : bool
- IsNullable : bool
- IsSubstitution : bool
- IsSystemAttribute : bool
- Name : string
- NetworkAttributeToSubstitute : NetworkAttribute
- Type : NetworkAttributeDataType



SubnetworkExportResultType
Features
Connectivity
ContainmentAndAttachment



ServiceSynchronizationType
SynchronousService
AsynchronousService



SubnetworkExportOptions

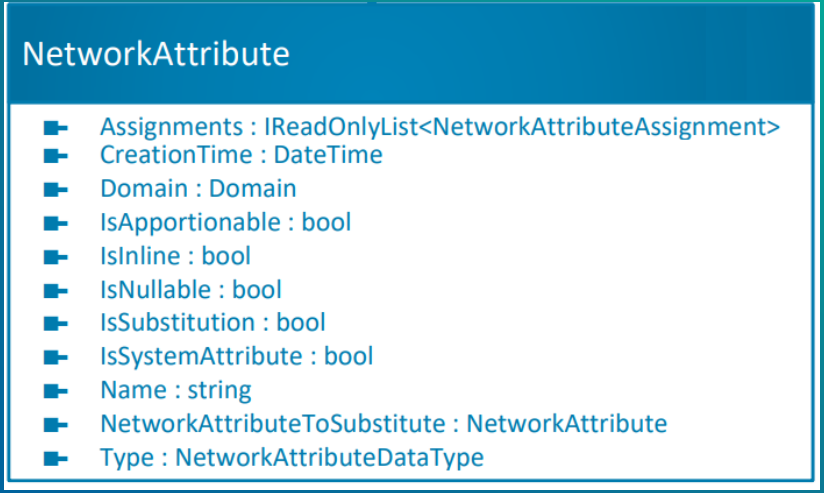
- IncludeDomainDescriptions : bool
- IncludeGeometry : bool
- ResultFieldsByNetworkSourceID : Dictionary<int, List<string>>
- ResultNetworkAttributes : List<NetworkAttribute>
- ServiceSynchronizationType : ServiceSynchronizationType
- SetAcknowledged : bool
- SubnetworkExportResultTypes : List<SubnetworkExportResultType>

▲ SubnetworkExportOptions()

Subnetwork as JSON

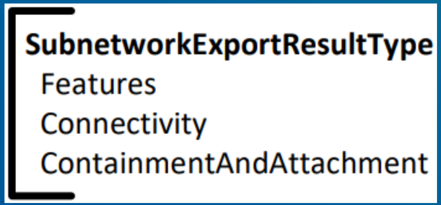
Export Options

- Defines the options to export a subnetwork
- `SubnetworkExportOptions` class
- `SetAcknowledged` property
- `True`
  - Updates the export ack date
  - To delete subnetwork controllers from the subnetworks table that have been removed as a subnetwork controller
- `False`
  - No controllers need to be deleted from the subnetworks table

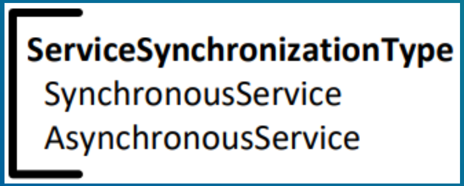


NetworkAttribute

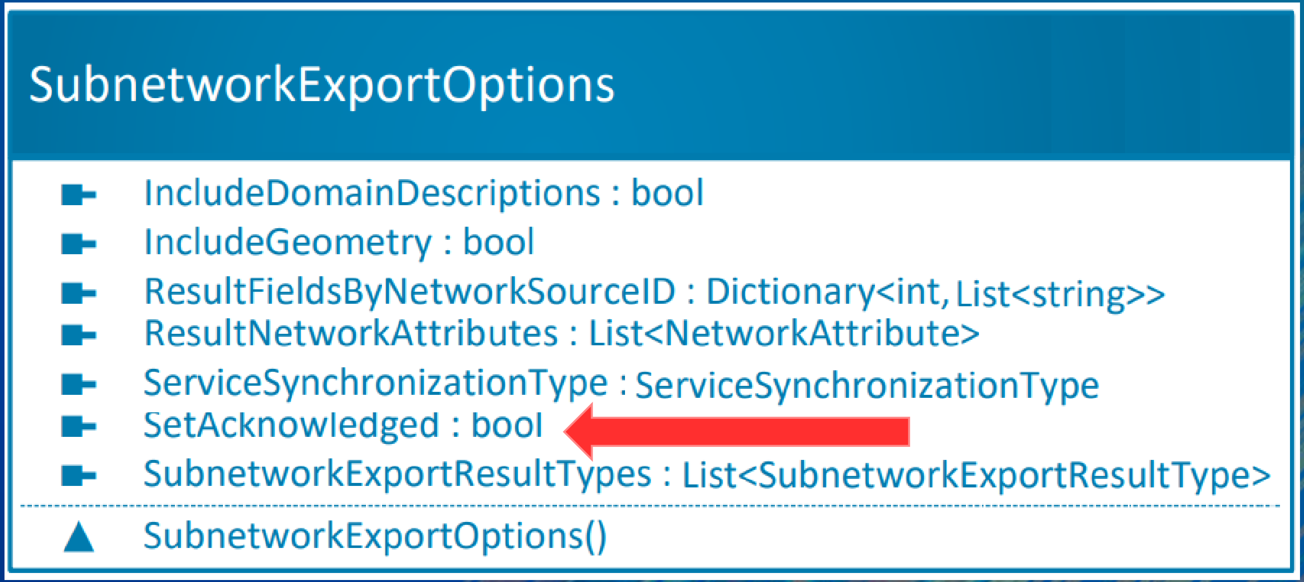
- Assignments : IReadOnlyList<NetworkAttributeAssignment>
- CreationTime : DateTime
- Domain : Domain
- IsApportionable : bool
- IsInline : bool
- IsNullable : bool
- IsSubstitution : bool
- IsSystemAttribute : bool
- Name : string
- NetworkAttributeToSubstitute : NetworkAttribute
- Type : NetworkAttributeDataType



SubnetworkExportResultType
Features
Connectivity
ContainmentAndAttachment



ServiceSynchronizationType
SynchronousService
AsynchronousService



SubnetworkExportOptions

- IncludeDomainDescriptions : bool
- IncludeGeometry : bool
- ResultFieldsByNetworkSourceID : Dictionary<int, List<string>>
- ResultNetworkAttributes : List<NetworkAttribute>
- ServiceSynchronizationType : ServiceSynchronizationType
- SetAcknowledged : bool
- SubnetworkExportResultTypes : List<SubnetworkExportResultType>

- SubnetworkExportOptions()

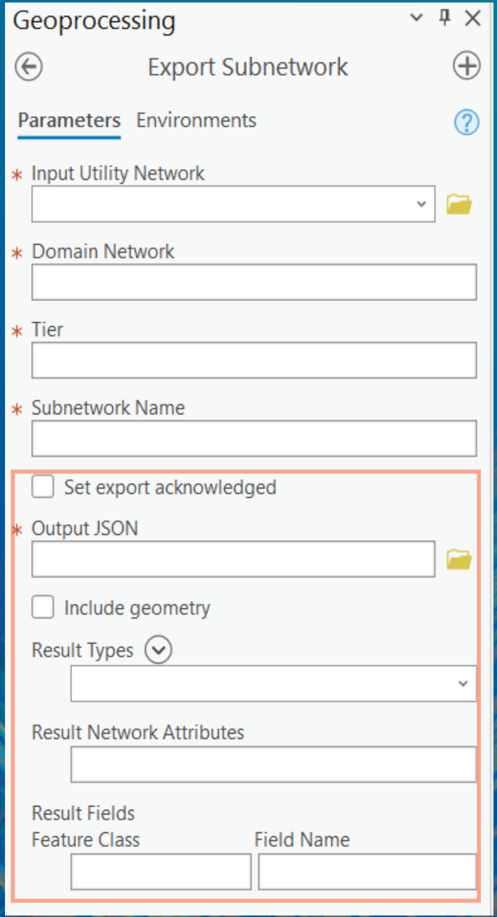
Subnetwork as JSON

Export to a JSON file

- `Subnetwork.Export(Uri outputJsonPath, SubnetworkExportOptions options)`

```
IReadOnlyList<NetworkAttribute> networkAttributes = utilityNetworkDefinition.GetNetworkAttributes();
IReadOnlyList<NetworkSource> networkSources = utilityNetworkDefinition.GetNetworkSources();
```

```
// Export options
SubnetworkExportOptions subnetworkExportOptions = new SubnetworkExportOptions()
{
  SetAcknowledged = false,
  IncludeDomainDescriptions = true,
  IncludeGeometry = true,
  ServiceSynchronizationType = ServiceSynchronizationType.Asynchronous,

  SubnetworkExportResultTypes = new List<SubnetworkExportResultType>()
  {
    SubnetworkExportResultType.Connectivity,
    SubnetworkExportResultType.Features
  },

  ResultNetworkAttributes = new List<NetworkAttribute>(networkAttributes),

  ResultFieldsByNetworkSourceID = new Dictionary<int, List<string>>()
    { { networkSources[0].ID, new List<string>() { "OBJECTID" } } }
};

// Export subnetwork
subnetwork.Export(exportResultJsonPath, subnetworkExportOptions);
```


Geoprocessing

Export Subnetwork

Parameters | Environments

* Input Utility Network

* Domain Network

* Tier

* Subnetwork Name

☐ Set export acknowledged

* Output JSON

☐ Include geometry

Result Types

Result Network Attributes

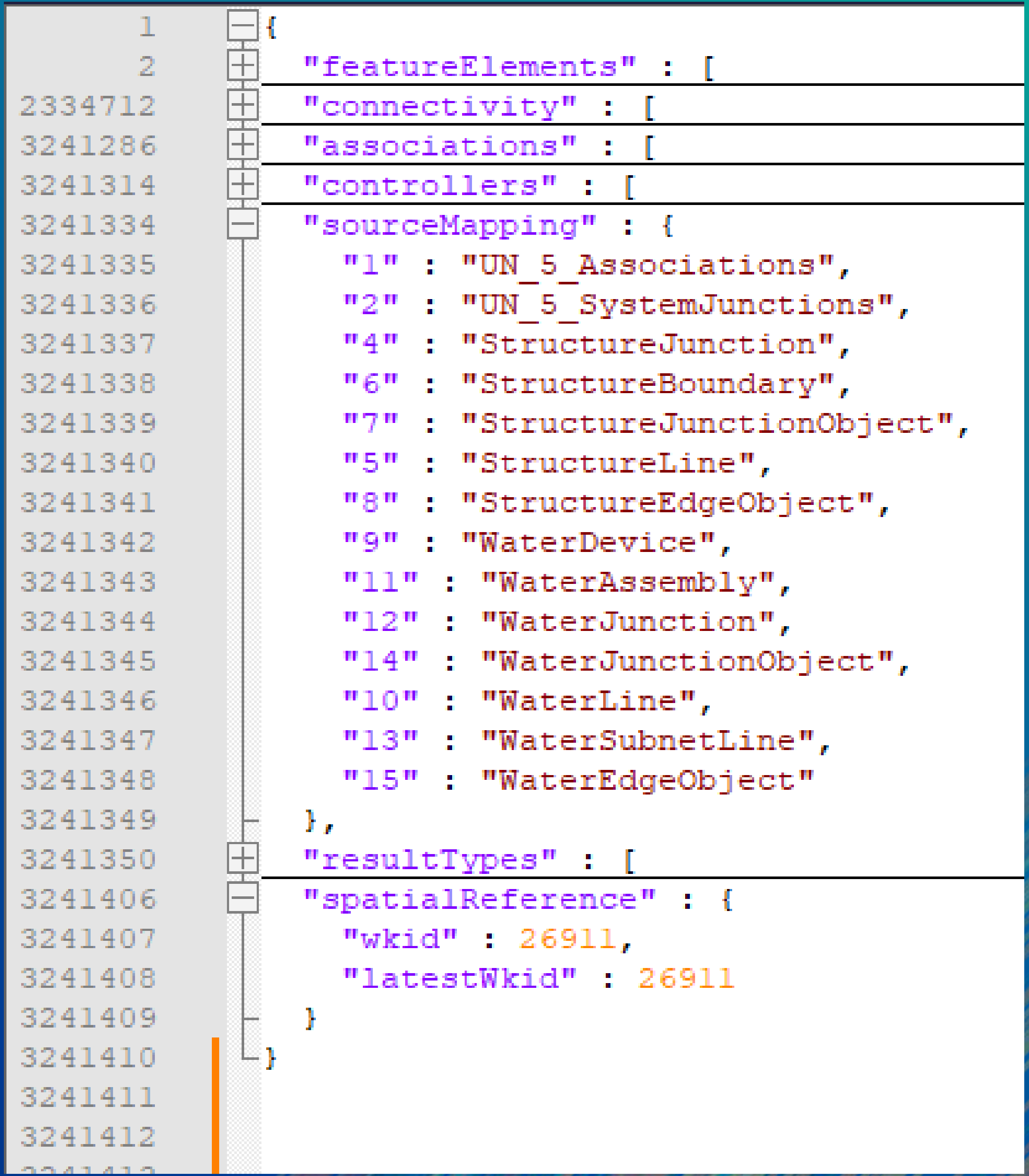
Result Fields

Feature Class	Field Name

JSON

Result

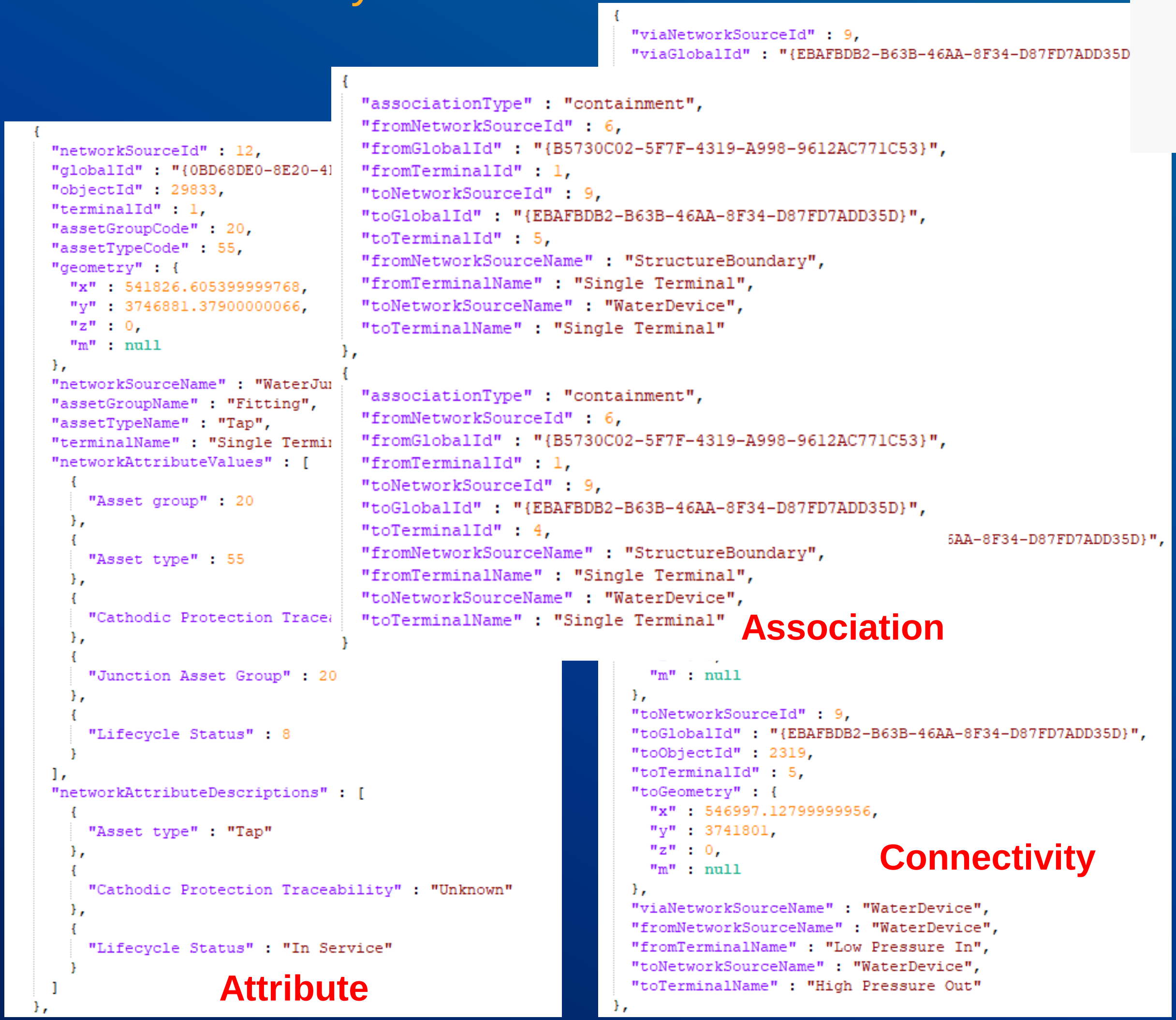
- Features
- Connectivity
- Associations
- Controllers
- Source mapping
- Results
- Very verbose
  - Network attribute
  - Result type setting



```
{
  "featureElements" : [
  "connectivity" : [
  "associations" : [
  "controllers" : [
  "sourceMapping" : {
    "1" : "UN_5_Associations",
    "2" : "UN_5_SystemJunctions",
    "4" : "StructureJunction",
    "6" : "StructureBoundary",
    "7" : "StructureJunctionObject",
    "5" : "StructureLine",
    "8" : "StructureEdgeObject",
    "9" : "WaterDevice",
    "11" : "WaterAssembly",
    "12" : "WaterJunction",
    "14" : "WaterJunctionObject",
    "10" : "WaterLine",
    "13" : "WaterSubnetLine",
    "15" : "WaterEdgeObject"
  },
  "resultTypes" : [
  "spatialReference" : {
    "wkid" : 26911,
    "latestWkid" : 26911
  }
}
```

Parse JSON

Network Analysis



```
{
  "viaNetworkSourceId" : 9,
  "viaGlobalId" : "{EBAFBDB2-B63B-46AA-8F34-D87FD7ADD35D
```

```
{
  "networkSourceId" : 12,
  "globalId" : "{0BD68DE0-8E20-4
  "objectId" : 29833,
  "terminalId" : 1,
  "assetGroupCode" : 20,
  "assetTypeCode" : 55,
  "geometry" : {
    "x" : 541826.605399999768,
    "y" : 3746881.37900000066,
    "z" : 0,
    "m" : null
  },
  "networkSourceName" : "WaterJu
  "assetGroupName" : "Fitting",
  "assetTypeName" : "Tap",
  "terminalName" : "Single Termi
  "networkAttributeValues" : [
    {
      "Asset group" : 20
    },
    {
      "Asset type" : 55
    },
    {
      "Cathodic Protection Trace
    },
    {
      "Junction Asset Group" : 20
    },
    {
      "Lifecycle Status" : 8
    }
  ],
  "networkAttributeDescriptions" : [
    {
      "Asset type" : "Tap"
    },
    {
      "Cathodic Protection Traceability" : "Unknown"
    },
    {
      "Lifecycle Status" : "In Service"
    }
  ]
},
```

Attribute

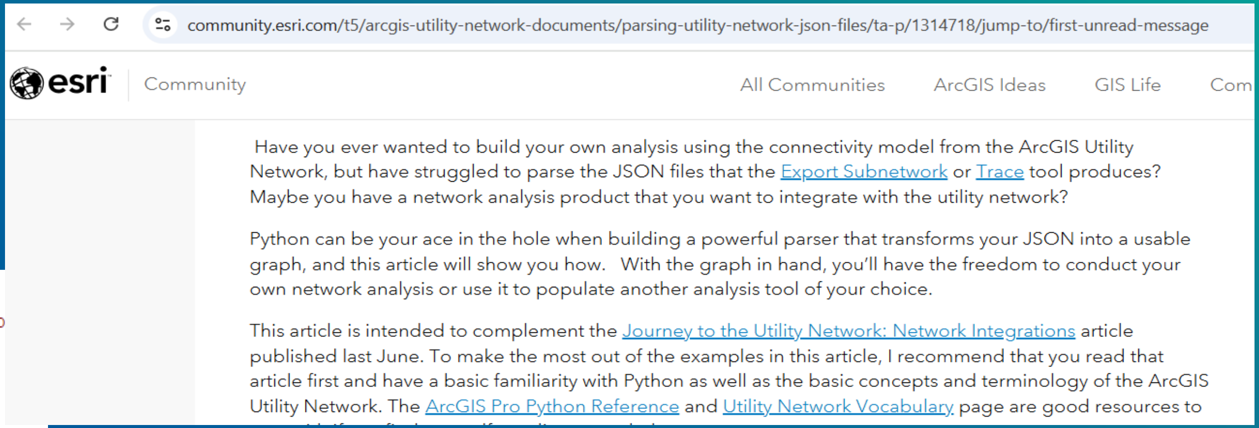
```
{
  "associationType" : "containment",
  "fromNetworkSourceId" : 6,
  "fromGlobalId" : "{B5730C02-5F7F-4319-A998-9612AC771C53}",
  "fromTerminalId" : 1,
  "toNetworkSourceId" : 9,
  "toGlobalId" : "{EBAFBDB2-B63B-46AA-8F34-D87FD7ADD35D}",
  "toTerminalId" : 5,
  "fromNetworkSourceName" : "StructureBoundary",
  "fromTerminalName" : "Single Terminal",
  "toNetworkSourceName" : "WaterDevice",
  "toTerminalName" : "Single Terminal"
},
{
  "associationType" : "containment",
  "fromNetworkSourceId" : 6,
  "fromGlobalId" : "{B5730C02-5F7F-4319-A998-9612AC771C53}",
  "fromTerminalId" : 1,
  "toNetworkSourceId" : 9,
  "toGlobalId" : "{EBAFBDB2-B63B-46AA-8F34-D87FD7ADD35D}",
  "toTerminalId" : 4,
  "fromNetworkSourceName" : "StructureBoundary",
  "fromTerminalName" : "Single Terminal",
  "toNetworkSourceName" : "WaterDevice",
  "toTerminalName" : "Single Terminal"
}
```

Association

```
5AA-8F34-D87FD7ADD35D}",
```

```
    "m" : null
  },
  "toNetworkSourceId" : 9,
  "toGlobalId" : "{EBAFBDB2-B63B-46AA-8F34-D87FD7ADD35D}",
  "toObjectId" : 2319,
  "toTerminalId" : 5,
  "toGeometry" : {
    "x" : 546997.12799999956,
    "y" : 3741801,
    "z" : 0,
    "m" : null
  },
  "viaNetworkSourceName" : "WaterDevice",
  "fromNetworkSourceName" : "WaterDevice",
  "fromTerminalName" : "Low Pressure In",
  "toNetworkSourceName" : "WaterDevice",
  "toTerminalName" : "High Pressure Out"
},
```

Connectivity



community.esri.com/t5/arcgis-utility-network-documents/parsing-utility-network-json-files/ta-p/1314718/jump-to/first-unread-message

Have you ever wanted to build your own analysis using the connectivity model from the ArcGIS Utility Network, but have struggled to parse the JSON files that the Export Subnetwork or Trace tool produces? Maybe you have a network analysis product that you want to integrate with the utility network?

Python can be your ace in the hole when building a powerful parser that transforms your JSON into a usable graph, and this article will show you how. With the graph in hand, you'll have the freedom to conduct your own network analysis or use it to populate another analysis tool of your choice.

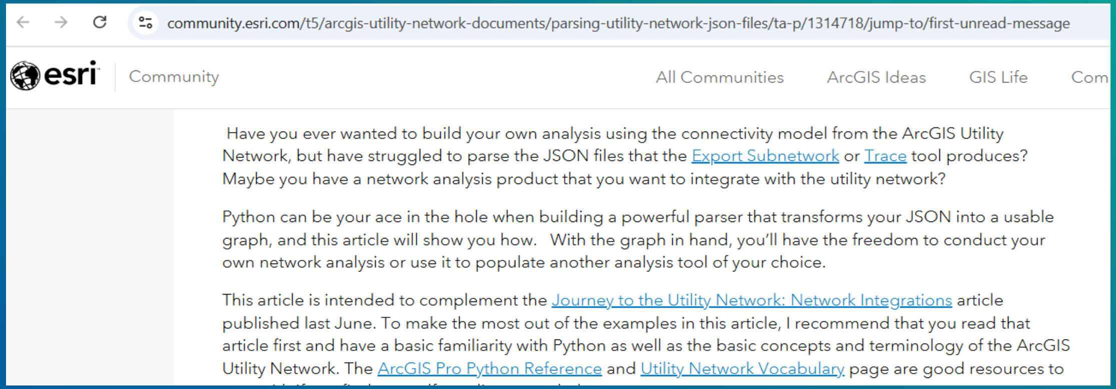
This article is intended to complement the Journey to the Utility Network: Network Integrations article published last June. To make the most out of the examples in this article, I recommend that you read that article first and have a basic familiarity with Python as well as the basic concepts and terminology of the ArcGIS Utility Network. The ArcGIS Pro Python Reference and Utility Network Vocabulary page are good resources to

Parsing Utility Network JSON Files

Parse JSON

Network Analysis

- NewtonSoft
  - `Newtonsoft.Json`
  - Mature
  - Wide range of configurability and compatibility
  - Easier serialization/deserialization of the JSON schema
- Microsoft
  - `System.Text.Json`
  - New and supposedly improved performance
  - Serializatio /Deserialization and support for complex JSON schema can be difficult
- Incrementally parse the file, element by element


community.esri.com/t5/arcgis-utility-network-documents/parsing-utility-network-json-files/ta-p/1314718/jump-to/first-unread-message
esri
Community
All Communities
ArcGIS Ideas
GIS Life
Com

Have you ever wanted to build your own analysis using the connectivity model from the ArcGIS Utility Network, but have struggled to parse the JSON files that the Export Subnetwork or Trace tool produces? Maybe you have a network analysis product that you want to integrate with the utility network?

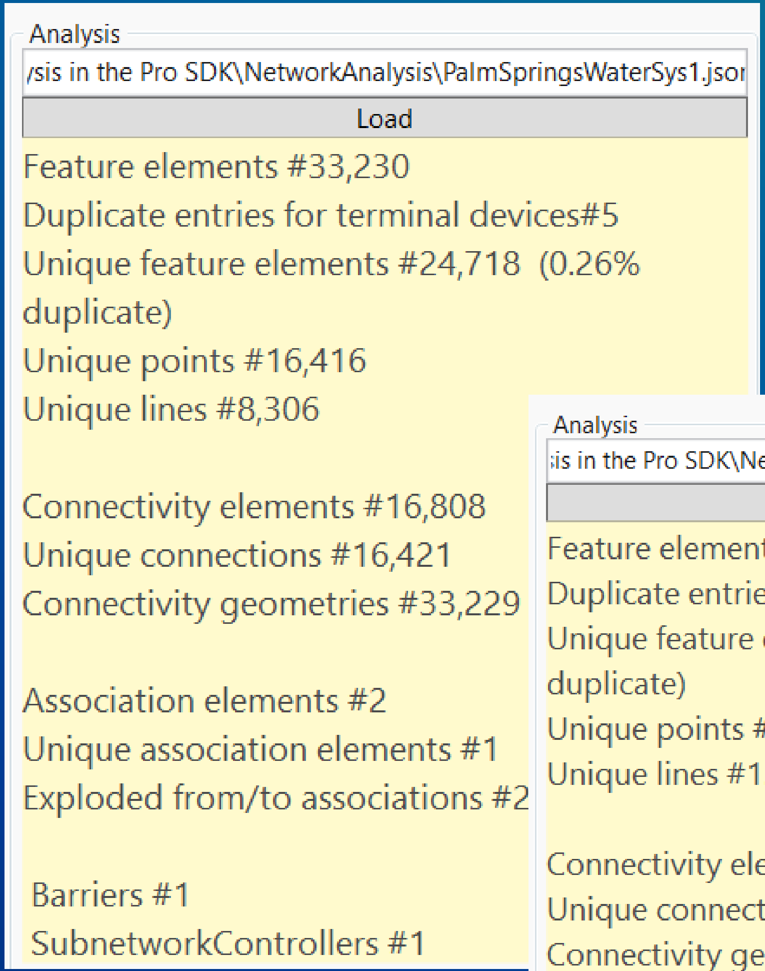
Python can be your ace in the hole when building a powerful parser that transforms your JSON into a usable graph, and this article will show you how. With the graph in hand, you'll have the freedom to conduct your own network analysis or use it to populate another analysis tool of your choice.

This article is intended to complement the Journey to the Utility Network: Network Integrations article published last June. To make the most out of the examples in this article, I recommend that you read that article first and have a basic familiarity with Python as well as the basic concepts and terminology of the ArcGIS Utility Network. The ArcGIS Pro Python Reference and Utility Network Vocabulary page are good resources to

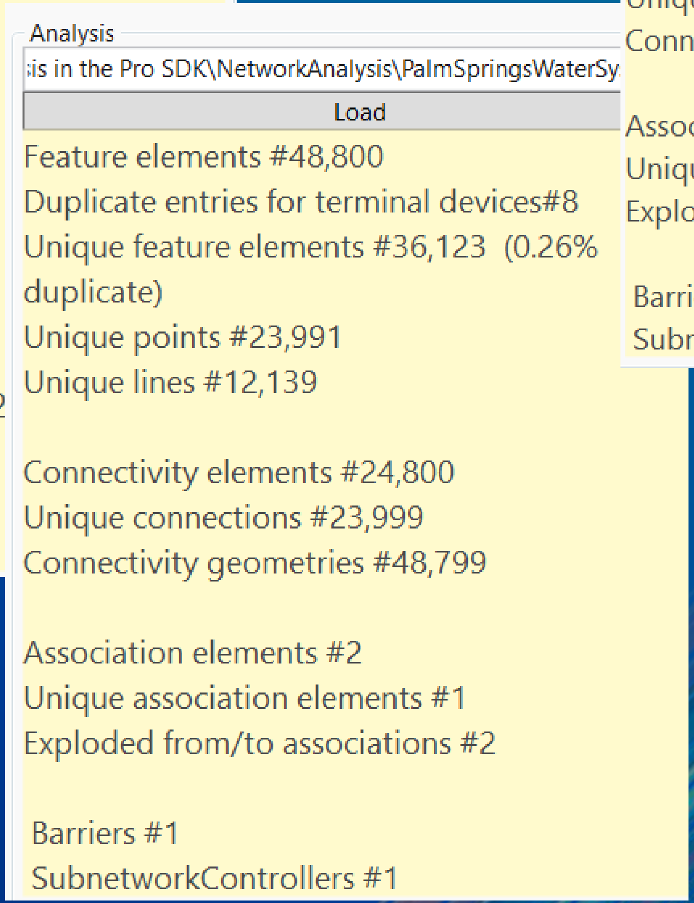
[Parsing Utility Network JSON Files]

Analysis Result

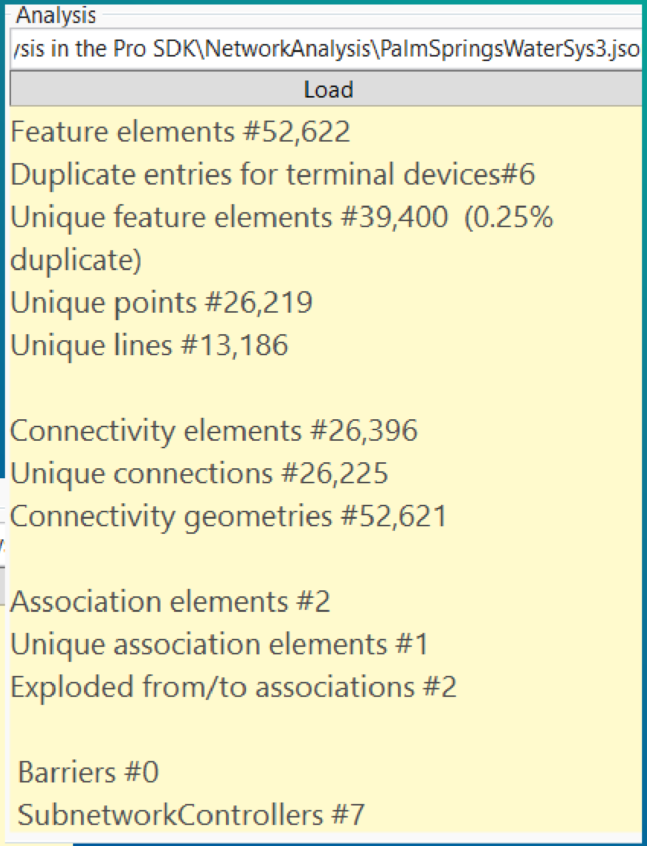
- Feature info
- Connectivity info
- Associations info
- Barrier
- Subnetwork controller
- Network graph



```
Analysis
/sis in the Pro SDK\NetworkAnalysis\PalmSpringsWaterSys1.jsor
Load
Feature elements #33,230
Duplicate entries for terminal devices#5
Unique feature elements #24,718  (0.26%
duplicate)
Unique points #16,416
Unique lines #8,306

Connectivity elements #16,808
Unique connections #16,421
Connectivity geometries #33,229

Association elements #2
Unique association elements #1
Exploded from/to associations #2

 Barriers #1
 SubnetworkControllers #1
```



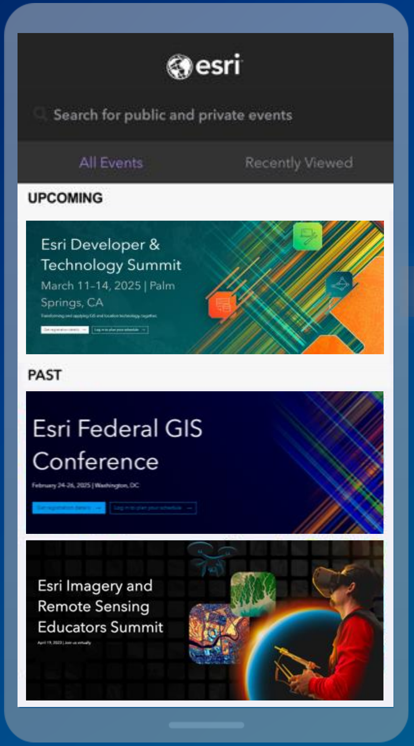
```
Analysis
;is in the Pro SDK\NetworkAnalysis\PalmSpringsWaterSy
Load
Feature elements #48,800
Duplicate entries for terminal devices#8
Unique feature elements #36,123  (0.26%
duplicate)
Unique points #23,991
Unique lines #12,139

Connectivity elements #24,800
Unique connections #23,999
Connectivity geometries #48,799

Association elements #2
Unique association elements #1
Exploded from/to associations #2

 Barriers #1
 SubnetworkControllers #1
```



```
Analysis
/sis in the Pro SDK\NetworkAnalysis\PalmSpringsWaterSys3.jso
Load
Feature elements #52,622
Duplicate entries for terminal devices#6
Unique feature elements #39,400  (0.25%
duplicate)
Unique points #26,219
Unique lines #13,186

Connectivity elements #26,396
Unique connections #26,225
Connectivity geometries #52,621

Association elements #2
Unique association elements #1
Exploded from/to associations #2

 Barriers #0
 SubnetworkControllers #7
```

Please share your feedback in the app

Download the Esri Events app and find your event



esri

Search for public and private events

All Events | Recently Viewed

UPCOMING

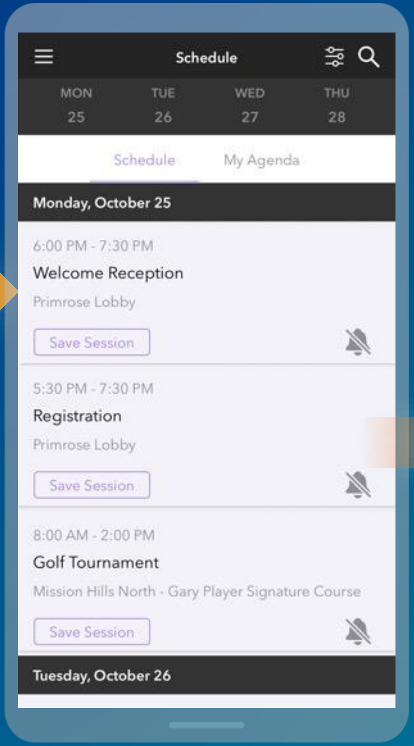
Esri Developer & Technology Summit
March 11-14, 2025 | Palm Springs, CA

PAST

Esri Federal GIS Conference

Esri Imagery and Remote Sensing Educators Summit

Select the session that you attended



Schedule

MON 25 | TUE 26 | WED 27 | THU 28

Schedule | My Agenda

Monday, October 25

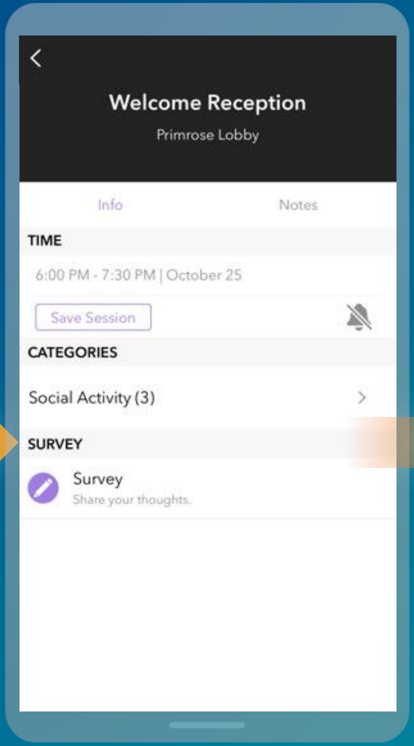
6:00 PM - 7:30 PM
Welcome Reception
Primrose Lobby
Save Session

5:30 PM - 7:30 PM
Registration
Primrose Lobby
Save Session

8:00 AM - 2:00 PM
Golf Tournament
Mission Hills North - Gary Player Signature Course
Save Session

Tuesday, October 26

Scroll down to "Survey"



Welcome Reception
Primrose Lobby

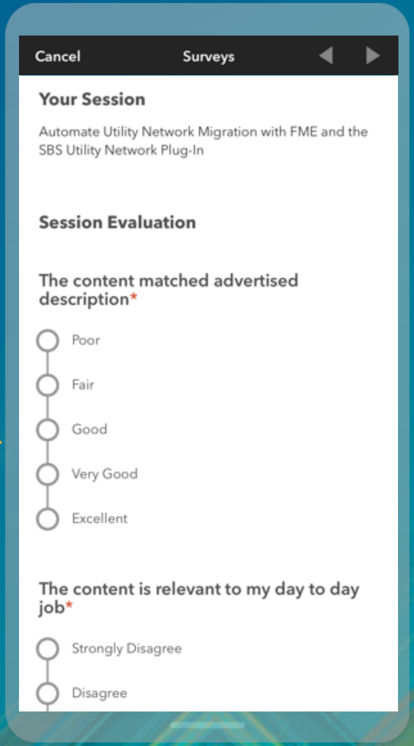
Info | Notes

TIME
6:00 PM - 7:30 PM | October 25
Save Session

CATEGORIES
Social Activity (3)

SURVEY
Survey
Share your thoughts.

Log in to access the survey



Cancel | Surveys

Your Session
Automate Utility Network Migration with FME and the SBS Utility Network Plug-In

Session Evaluation

The content matched advertised description*
- Poor
- Fair
- Good
- Very Good
- Excellent

The content is relevant to my day to day job*
- Strongly Disagree
- Disagree

Connect with us on Social

Join the Conversation using #EsriDevTech2025

- x.com/EsriDevs
- x.com/EsriDevEvents
- youtube.com/@EsriDevs
- links.esri.com/DevVideos
- github.com/Esri
- github.com/EsriDevEvents
- links.esri.com/EsriDevCommunity

esri®

THE SCIENCE OF WHERE®