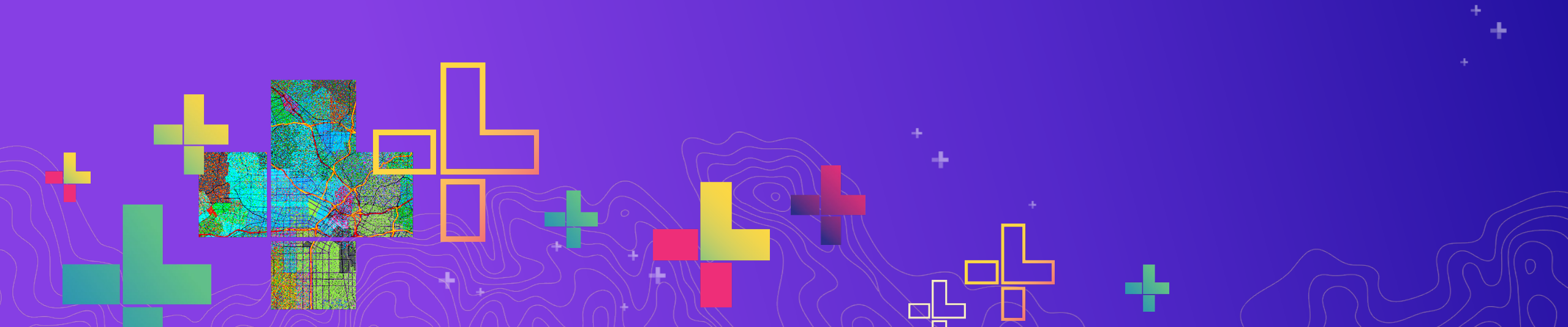




ArcGIS Pro SDK for .NET: An Overview of the Utility Network Management API

Rich Ruh, David Crawford

2020 ESRI DEVELOPER SUMMIT | Palm Springs, CA



Today's Agenda

- Getting Started
- A Survey of the Utility Network APIs
- What's New?
- The Road Ahead
- Learning More



Getting Started



https://pro.arcgis.com/en/pro-app/sdk/

Search for: "ArcGIS Pro SDK"

The screenshot shows a web browser window displaying the ArcGIS Pro SDK for Microsoft .NET page. The browser's address bar shows the URL <https://pro.arcgis.com/en/pro-app/sdk/>. The page features a navigation menu with links for Home, Get Started, Help, Tool Reference, Python, SDK, and Community. The main heading is "ArcGIS Pro SDK for Microsoft .NET". Below the heading, a paragraph states: "Extend ArcGIS Pro using the ArcGIS Pro SDK for Microsoft .NET. Develop add-ins and solution configurations to create a custom Pro UI and user experience for your organization. Download within Visual Studio or at My Esri." A secondary navigation bar includes links for "Released Version: 2.1 (January 2018)", "Installation", "What's new in 2.1", "API Reference", "Community Samples", and "Documentation". The "Get started with ArcGIS DevLabs" section contains five cards, each with a title, description, estimated time, and a "Start the lab" button:

- Build your first add-in**: You will learn how to create and run a basic ArcGIS Pro add-in. Estimated time: 15 minutes.
- Build your first configuration**: You will learn how to create and run a basic ArcGIS Pro solution configuration. Estimated time: 15 minutes.
- Build a map identification tool**: You will learn how to create an ArcGIS Pro add-in with a custom map identification tool. Estimated time: 15 minutes.
- Build a feature construction tool**: You will learn how to create an ArcGIS Pro feature construction tool. Estimated time: 15 minutes.
- Prepare your data for offline use**: Create a mobile map package (MMPK) in ArcGIS Pro that can be used by mobile SDKs for offline data access. Estimated time: 15 minutes.

Overview

- The utility network C# SDK is a managed .NET SDK that provides access to the utility network
- It is an object-oriented SDK that aligns with modern C# practices and existing frameworks
- It adheres to the principles and architecture of the general Pro SDK
- This presentation assumes a basic understanding of the utility network information model



Architectural Topics

- DML-only (Data Manipulation Language)
- Threading



Architecture: DML-only (Data Manipulation Language)

- The utility network API is a **DML-only (Data Manipulation Language) API**
 - Schema creation and modification operations such as creating domain networks, adding and deleting rules, etc., need to be performed using Python
 - This is in alignment with the rest of the [Geodatabase API](#)
 - Python can be called from C# by using the [Geoprocessing API](#)

```
var args = Geoprocessing.MakeValueArray(utilityNetworkPath, @"ALL", @"rules.csv");  
var result = Geoprocessing.ExecuteToolAsync("un.ImportRules", args);
```

Architecture: Threading

- Almost all of the methods in the utility network API should be called on the MCT (Main CIM Thread)
- Read [Working with multithreading in ArcGIS Pro](#) to learn more

```
Task t = QueuedTask.Run(() =>
{
    //put utility network code here
});
```


Architecture: Namespaces and Extension Methods

- The main items of the utility network are included within the `ArcGIS.Core.Data.UtilityNetwork` namespace
 - Tracing items are included within `ArcGIS.Core.Data.UtilityNetwork.Trace`
 - Network diagram items are included within `ArcGIS.Core.Data.UtilityNetwork.NetworkDiagrams`
- There are some cases where the objects in `ArcGIS.Core.Data` need to integrate with ArcGIS Pro at a higher level in the architectural stack
- In these cases, C# extension methods are provided
 - To use these extension methods:
 - Add a reference to `ArcGIS.Desktop.Extensions` to your solution
 - Add `using ArcGIS.Core.Data.UtilityNetwork.Extensions` to the top of your source files
 - These steps allow these extension methods to appear as if they were regular methods on utility network classes
 - These extension methods are intended for use in ArcGIS Pro Add-ins, and cannot be used with CoreHost applications

Other Ways to Access the Utility Network

- In addition to the ArcGIS Pro Managed SDK, there are other ways to program against a utility network:
 - Geoprocessing models and Python scripts
 - Directly coding against the REST APIs

```
# Update subnetworks
arcpy.AddMessage("Update subnetworks")
arcpy.UpdateSubnetwork_un(utilityNetwork, domainNetworkName, "Subtransmission", "ALL_SUBNETWORKS_IN_TIER")
arcpy.UpdateSubnetwork_un(utilityNetwork, domainNetworkName, "Medium Voltage", "ALL_SUBNETWORKS_IN_TIER")
arcpy.UpdateSubnetwork_un(utilityNetwork, domainNetworkName, "Low Voltage Mesh", "ALL_SUBNETWORKS_IN_TIER")
arcpy.AddMessage("Finished updating subnetworks")
```

validateNetworkTopology

POST only

Validating the network topology for a utility network maintains consistency between feature editing space and network topology space. Validating a network topology may include all or a subset of the dirty areas present in the network.

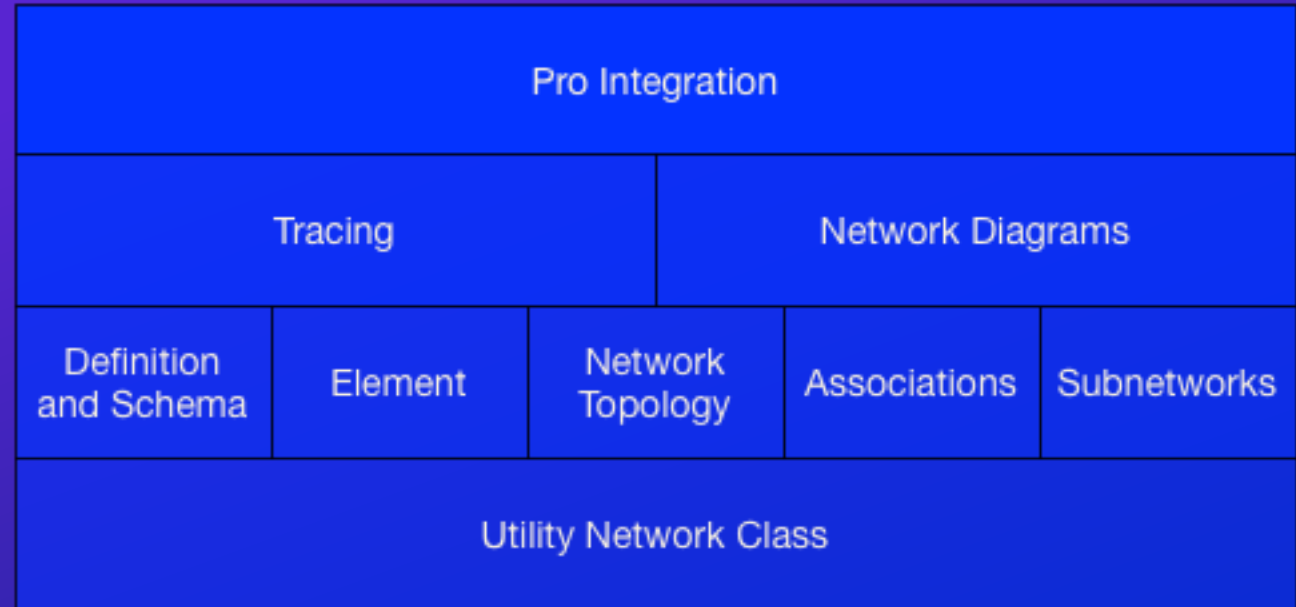
Parameter	Details
f	Description: Optional parameter representing the output format of the response (default is JSON).
gdbVersion	Description: The name of the GDB version. Syntax: gdbVersion=<version>
sessionId	Description: The token (guid) used to lock the version. Syntax : sessionId=<guid>
validateArea	Description: The envelope of the area to validate.

A Survey of the Utility Network APIs



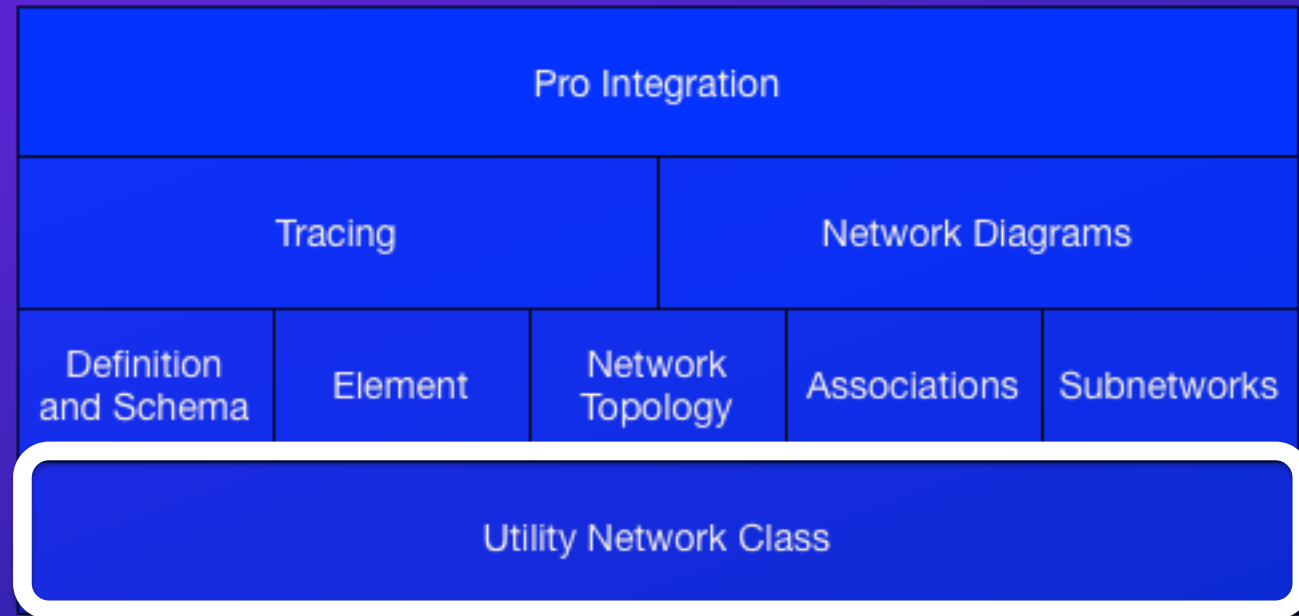
Organization of the Utility Network API

- The API can be logically divided into nine different sections
- The diagram at right provides a functional organization of the API
 - Strictly speaking, the API is a collection of classes
 - Not a layered architecture



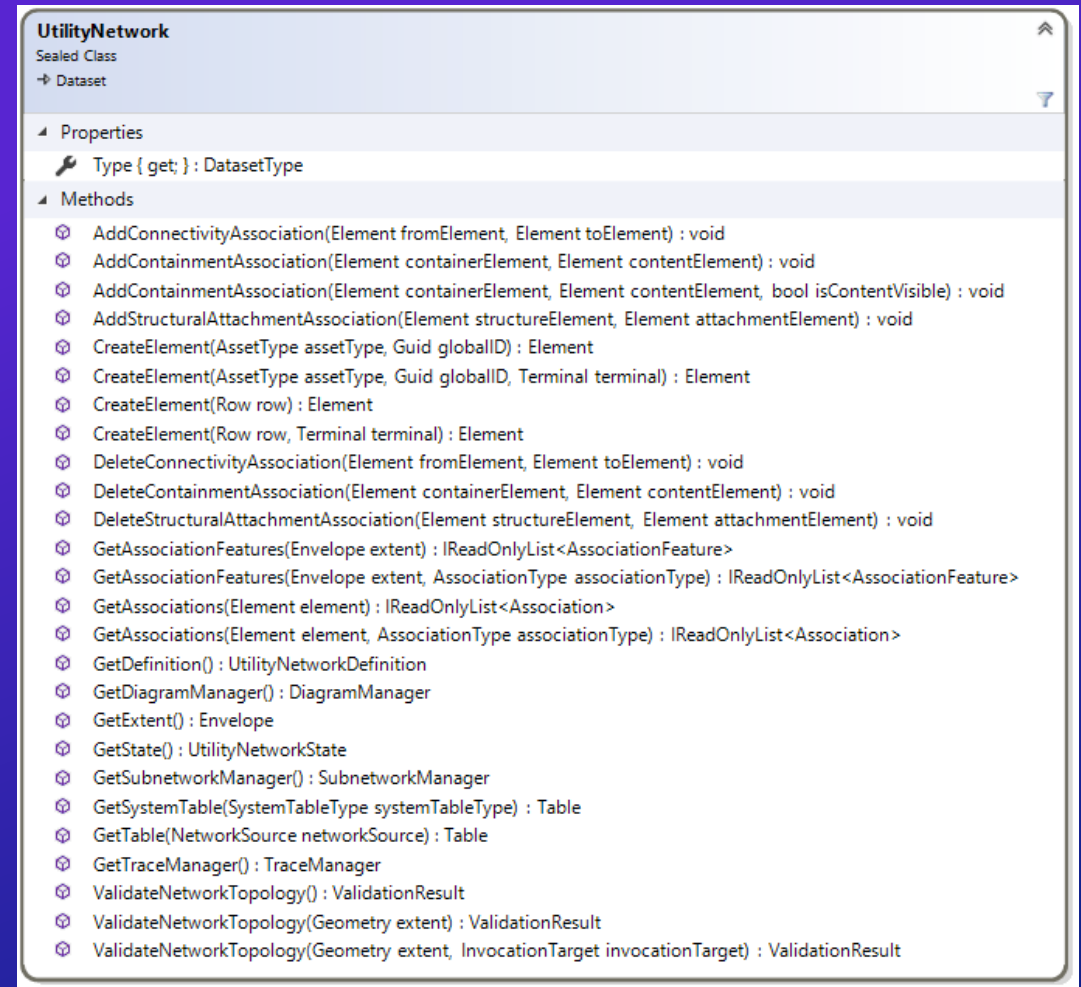
Organization of the Utility Network API

- The Utility Network Class is the root object that provides access to the utility network API



The UtilityNetwork Class

- Serves as the central hub of the utility network API
- Can be obtained from
 - A geodatabase
 - `Geodatabase.OpenDataset<UtilityNetwork>(string datasetName) : UtilityNetwork`
 - A feature class or table
 - `Table.GetControllerDataset() : IReadOnlyList<Dataset>`
 - A utility network layer
 - `GetUtilityNetwork() : UtilityNetwork`



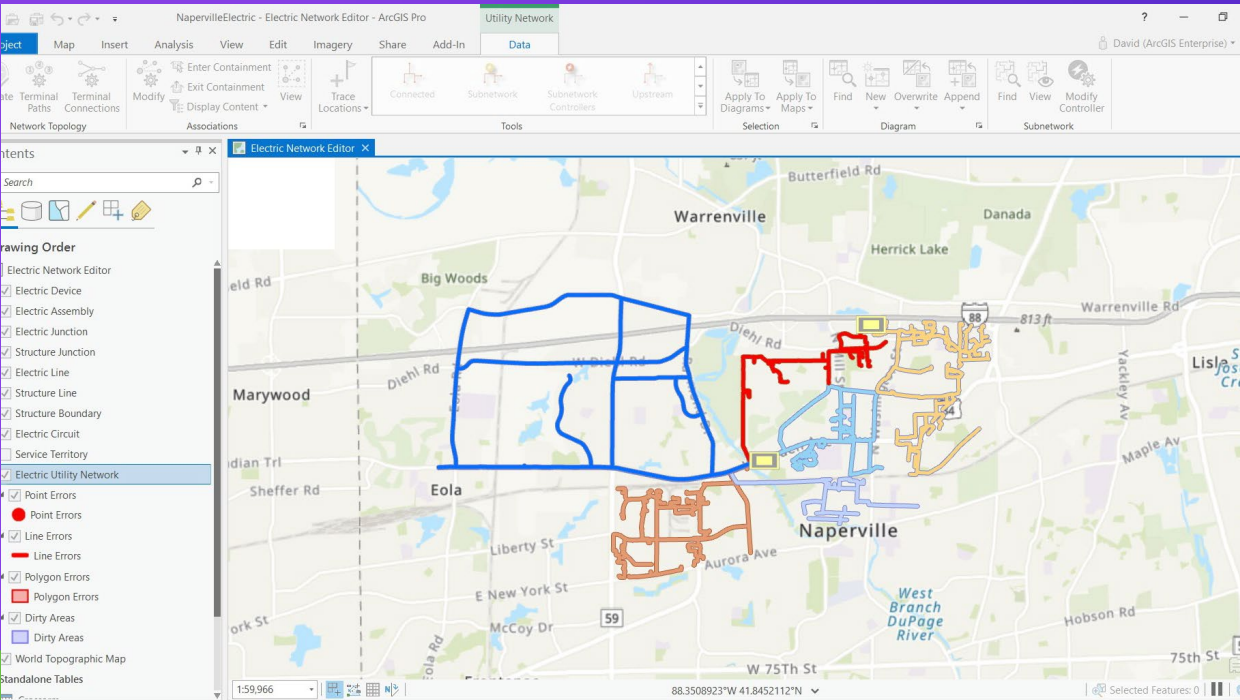
The screenshot shows the documentation for the `UtilityNetwork` class. It is identified as a "Sealed Class" and is associated with the "Dataset" namespace. The documentation is organized into two main sections: "Properties" and "Methods".

Properties:

- `Type { get; } : DatasetType`

Methods:

- `AddConnectivityAssociation(Element fromElement, Element toElement) : void`
- `AddContainmentAssociation(Element containerElement, Element contentElement) : void`
- `AddContainmentAssociation(Element containerElement, Element contentElement, bool isContentVisible) : void`
- `AddStructuralAttachmentAssociation(Element structureElement, Element attachmentElement) : void`
- `CreateElement(AssetType assetType, Guid globalID) : Element`
- `CreateElement(AssetType assetType, Guid globalID, Terminal terminal) : Element`
- `CreateElement(Row row) : Element`
- `CreateElement(Row row, Terminal terminal) : Element`
- `DeleteConnectivityAssociation(Element fromElement, Element toElement) : void`
- `DeleteContainmentAssociation(Element containerElement, Element contentElement) : void`
- `DeleteStructuralAttachmentAssociation(Element structureElement, Element attachmentElement) : void`
- `GetAssociationFeatures(Envelope extent) : IReadOnlyList<AssociationFeature>`
- `GetAssociationFeatures(Envelope extent, AssociationType associationType) : IReadOnlyList<AssociationFeature>`
- `GetAssociations(Element element) : IReadOnlyList<Association>`
- `GetAssociations(Element element, AssociationType associationType) : IReadOnlyList<Association>`
- `GetDefinition() : UtilityNetworkDefinition`
- `GetDiagramManager() : DiagramManager`
- `GetExtent() : Envelope`
- `GetState() : UtilityNetworkState`
- `GetSubnetworkManager() : SubnetworkManager`
- `GetSystemTable(SystemTableType systemTableType) : Table`
- `GetTable(NetworkSource networkSource) : Table`
- `GetTraceManager() : TraceManager`
- `ValidateNetworkTopology() : ValidationResult`
- `ValidateNetworkTopology(Geometry extent) : ValidationResult`
- `ValidateNetworkTopology(Geometry extent, InvocationTarget invocationTarget) : ValidationResult`

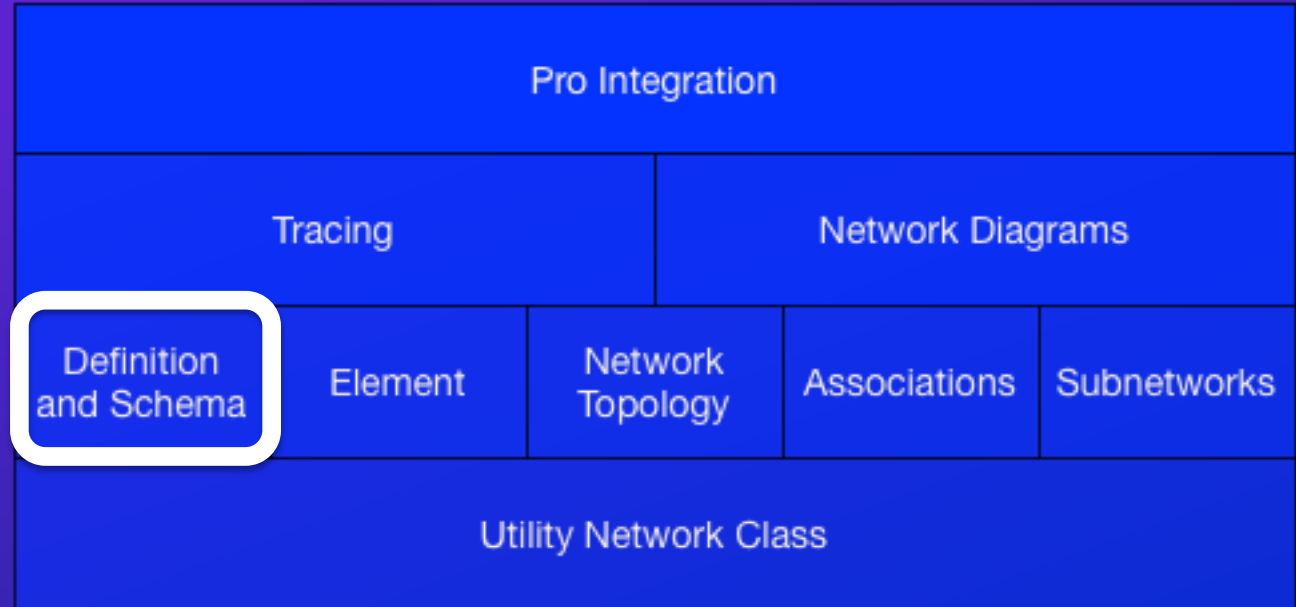


Obtaining a Utility Network Object

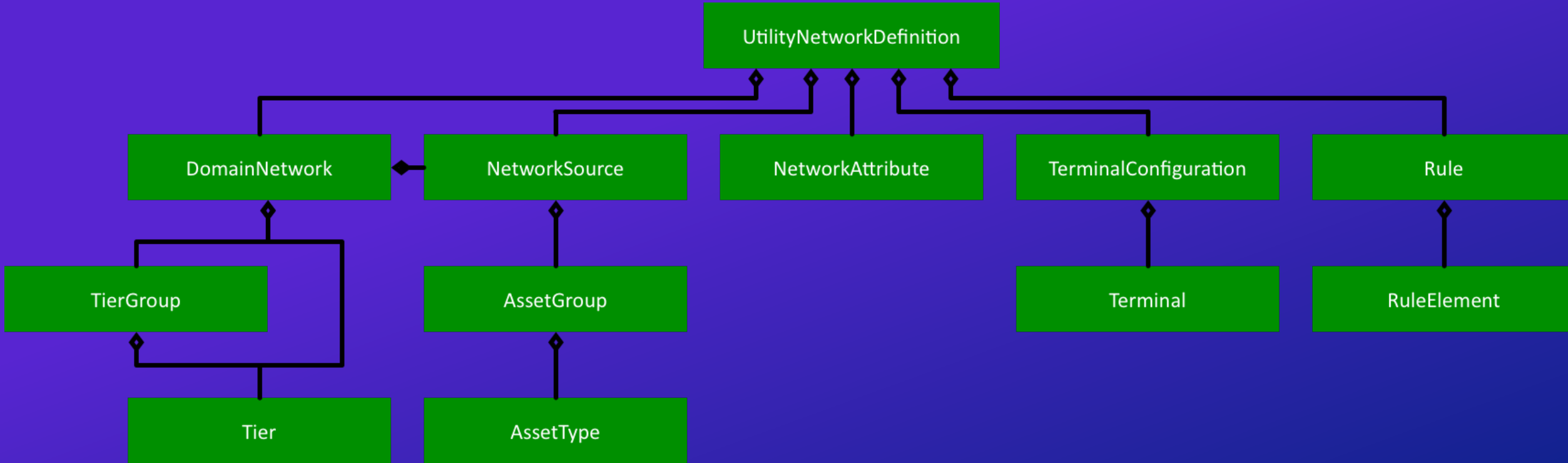
David Crawford

Organization of the Utility Network API

- Definition and Schema describes the classes and methods that provide information about the utility network schema



Overview of Definition and Schema Classes

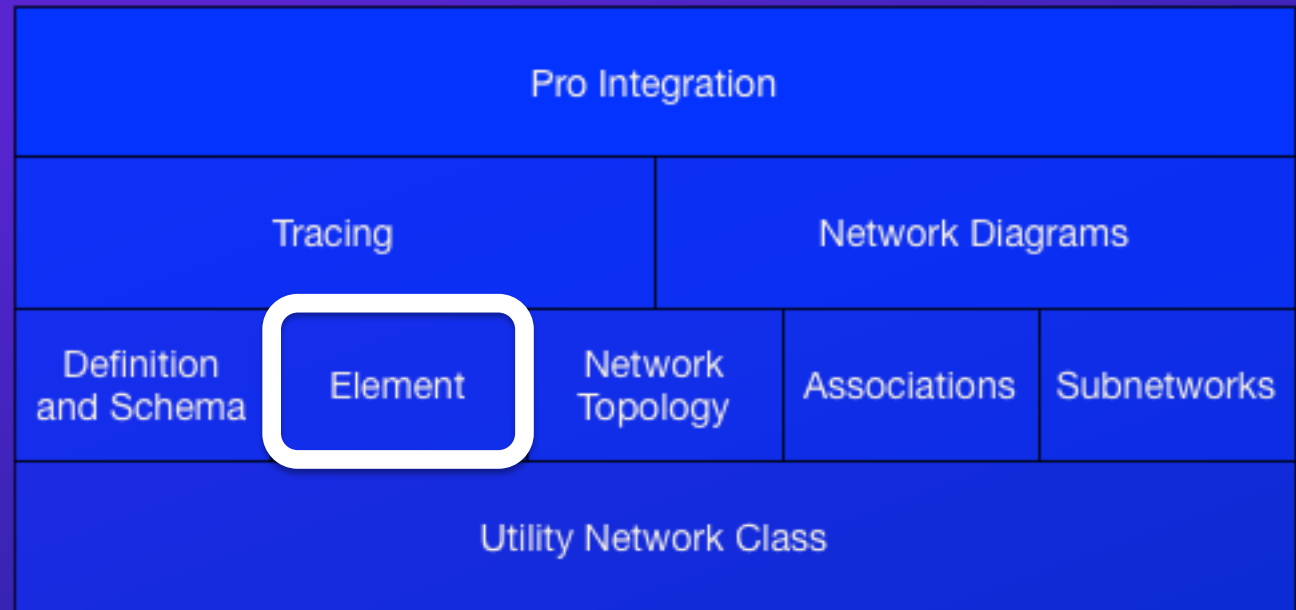


+ These classes provide read-only access to schema information

+ These classes are value objects that are derived from information cached with the feature service

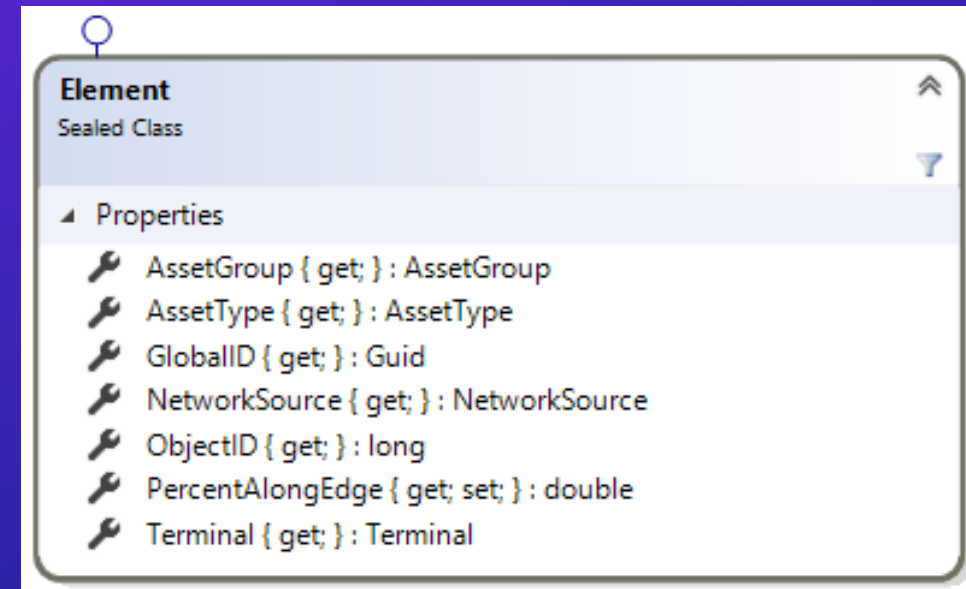
Organization of the Utility Network API

- Element covers the basic encapsulation of a row in the utility network API



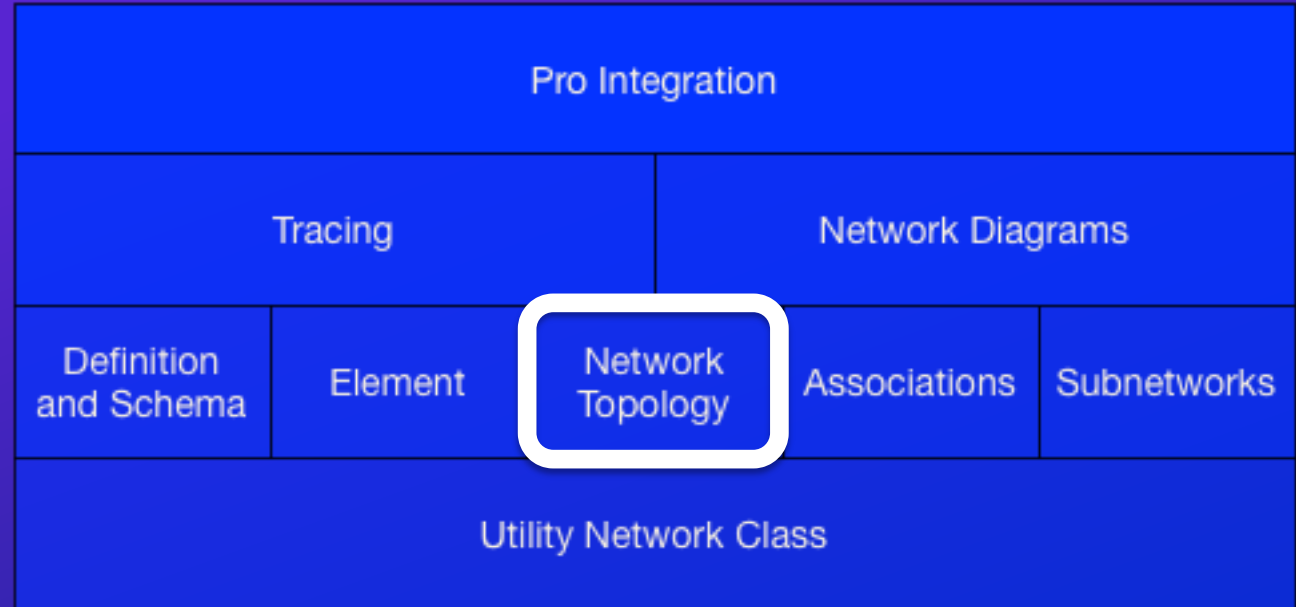
Elements

- The `Element` class represents a row inside a utility network, *plus* a terminal or percent along edge (if applicable)
- Used throughout the API:
 - Elements are used to create and delete associations
 - Elements specify starting points and barriers for use with tracing
 - Elements are returned as results from traces
 - Etc.
- Created using `CreateElement()` factory methods on the `UtilityNetwork` class



Organization of the Utility Network API

- Network Topology covers routines that query the topological index



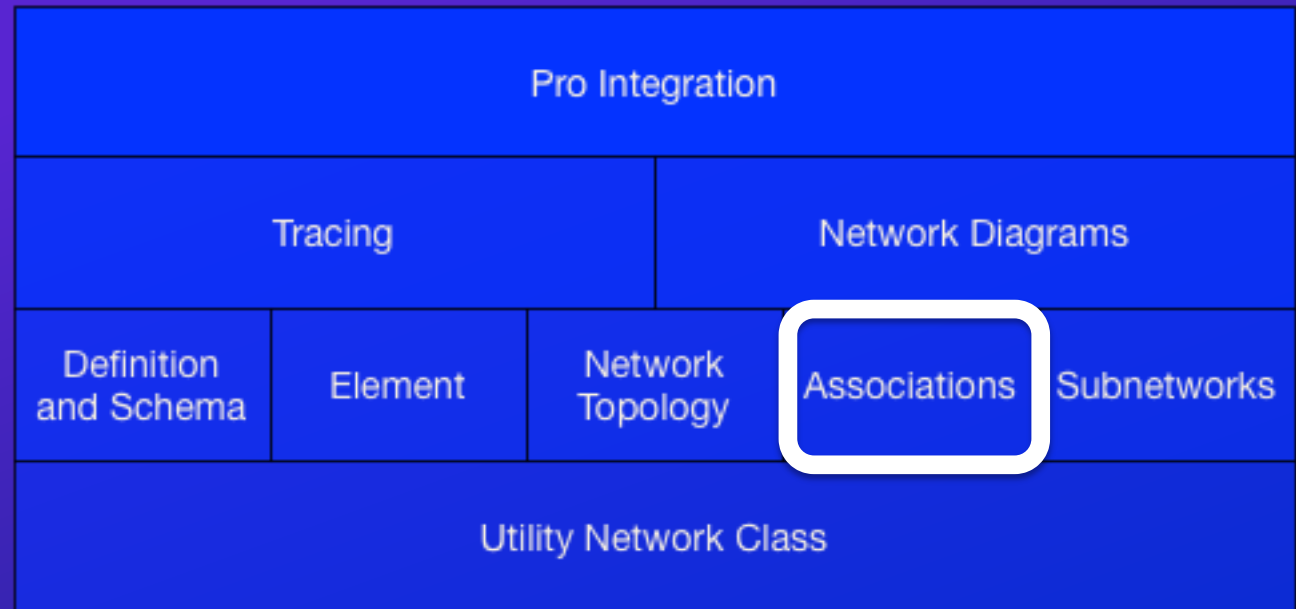
Utility Network Topology

- The network topology stores connectivity, containment, and attachment information used by the utility network to facilitate fast network traversal/analytical operations
- Network topology is constructed from
 - Geometric coincidence *and...*
 - Associations *in combination with...*
 - A powerful rules engine
- Topology is updated and validated with the `ValidateNetworkTopologyInEditOperation()` extension method on the `UtilityNetwork` class
- Access to fine-grained topology is not provided



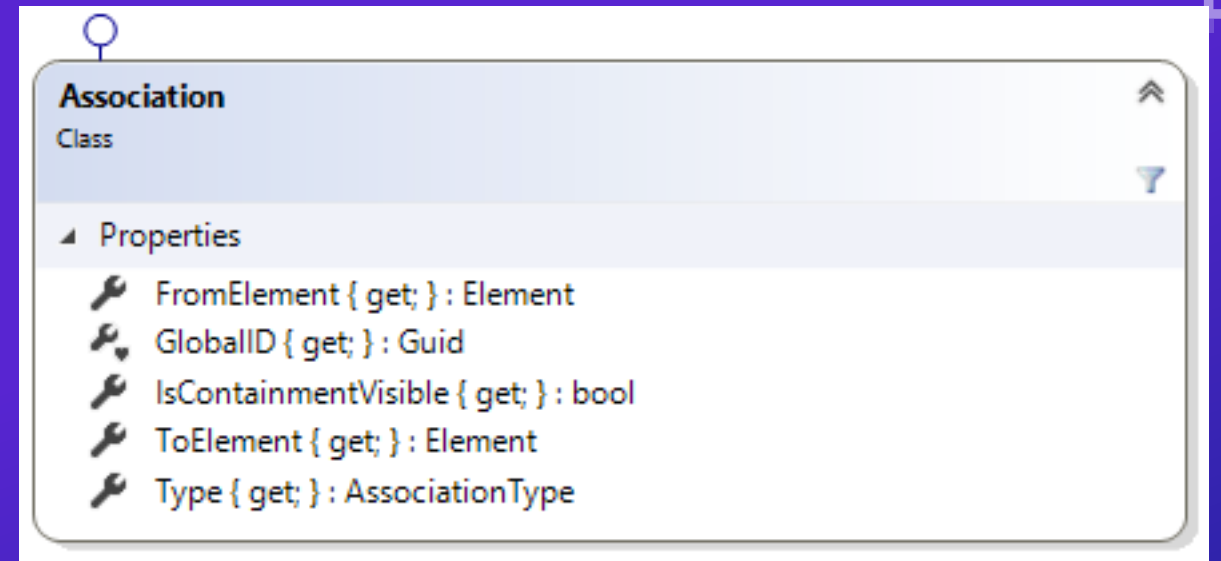
Organization of the Utility Network API

- Associations covers routines that query and edit associations between utility network rows



Associations

- Association Types
 - Connectivity
 - Containment
 - Structural Attachment
- Query
 - Get associations for a given Element
- Edit
 - Create or delete associations

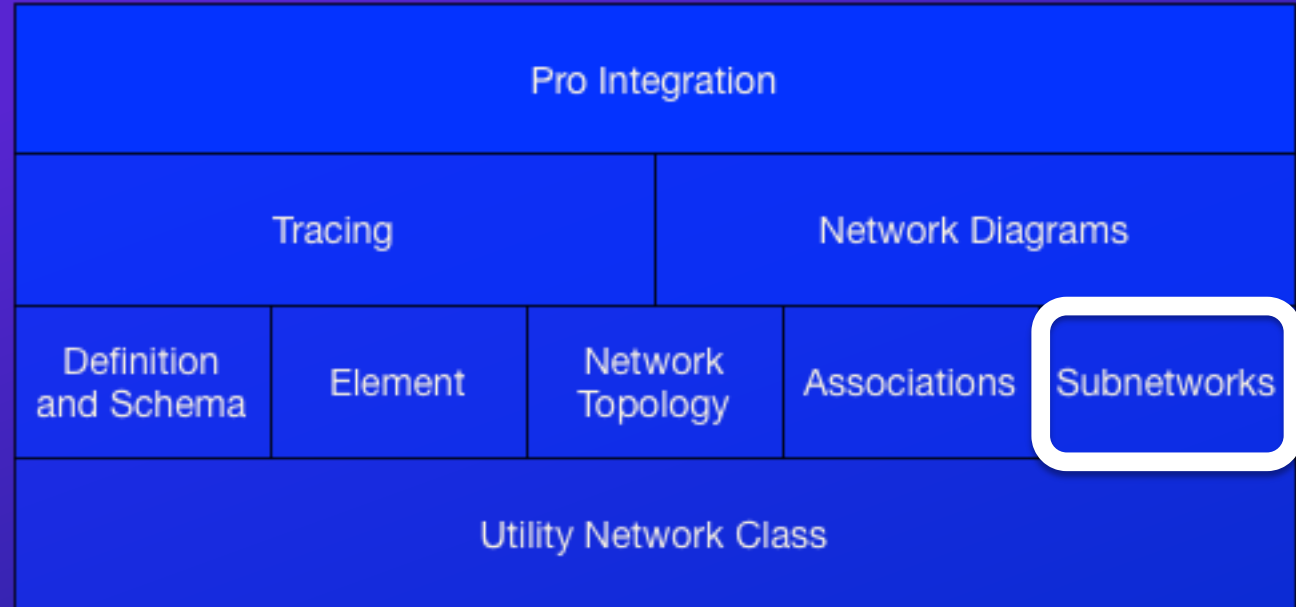


The screenshot displays a software interface for the 'Association' class. The class is identified as 'Class' and has a search icon at the top left. The 'Properties' section is expanded, showing five properties, each with a key icon:

- FromElement { get; } : Element
- GlobalID { get; } : Guid
- IsContainmentVisible { get; } : bool
- ToElement { get; } : Element
- Type { get; } : AssociationType

Organization of the Utility Network API

- Subnetworks provides classes and routines to query and edit utility network subnetworks



Subnetworks

- The management of subnetworks allows organizations to optimize the delivery of resources and track the status of a network
- A single subnetwork can be used to model such things as a circuit in electric networks, and a zone in gas and water networks
- The `SubnetworkManager` is a class that contains a collection of subnetwork management routines



Subnetwork Manager

- Query
 - `GetSubnetworks()` allows retrieval of Subnetworks based on their state
- Edit
 - Add and remove controllers with `EnableControllerInEditOperation()` and `DisableControllerInEditOperation()`
 - Extension methods
 - Update all subnetworks with `UpdateAllSubnetworks()`

```
SubnetworkManager
Sealed Class
↳ CoreObjectsBase

Methods
  DisableController(Element device) : void
  EnableController(Tier tier, Element device, string subnetworkName, string controllerName, string description, string notes) : Subnetwork
  GetSubnetworks(Tier tier, SubnetworkStates subnetworkStates) : IReadOnlyList<Subnetwork>
  UpdateAllSubnetworks(Tier tier, bool continueOnFailure) : void
```

The Subnetwork Class

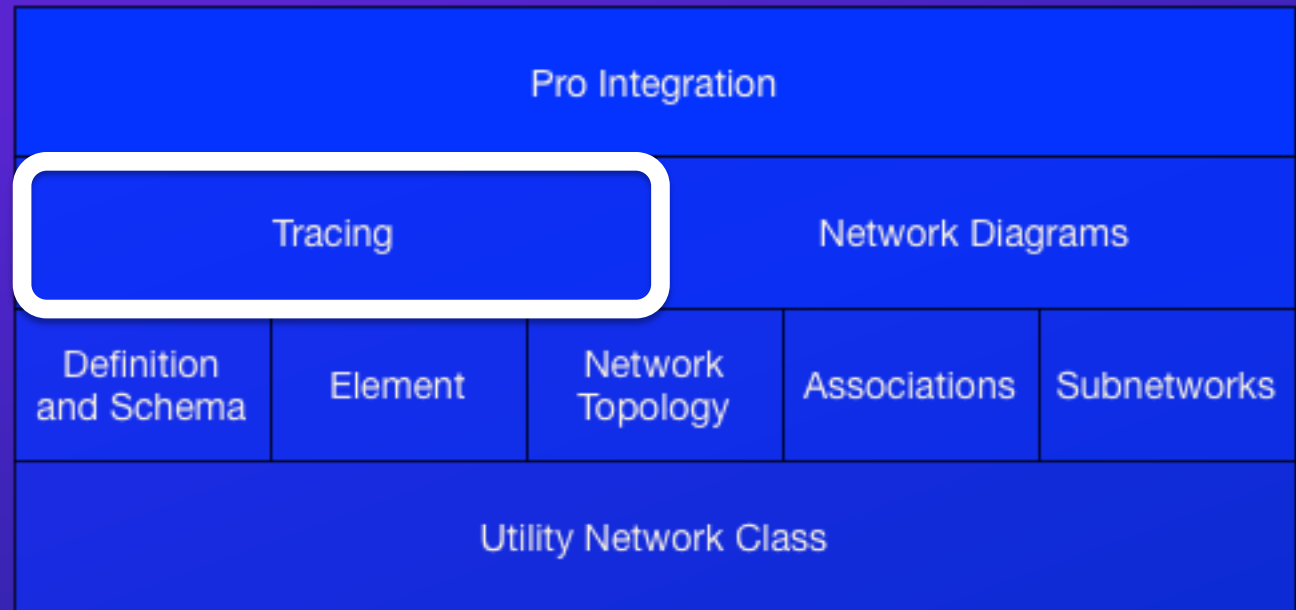
- Represents a subnetwork
- Update a subnetwork
 - Redraw the map after!
 - `MapView.Redraw(true)`
- Fetch the SubnetLine feature
- Return set of system network diagrams associated with this subnetwork

Subnetwork
Sealed Class

- ▲ Properties
 - 🔑 Name { get; } : string
 - 🔑 Tier { get; } : Tier
- ▲ Methods
 - 🔒 GetControllers() : IReadOnlyList<SubnetworkController>
 - 🔒 GetLastAcknowledgedExport() : DateTime
 - 🔒 GetLastUpdate() : DateTime
 - 🔒 GetLineFeature() : Feature
 - 🔒 GetNetworkDiagrams() : IReadOnlyList<NetworkDiagram>
 - 🔒 GetState() : SubnetworkStates
 - 🔒 Update() : void
 - 🔒 Update(TraceConfiguration traceConfiguration) : void

Organization of the Utility Network API

- Tracing provides tracing functionality
- Tracing entails assembling a subset of utility network elements that meet a specified criteria
- Tracing uses network data to provide business value to utilities
 - Answers questions and solves problems about the current state of the network
 - Helps design future facilities
 - Helps organize business practices



Components of a Trace

- Different kinds of traces are implemented with Tracer objects

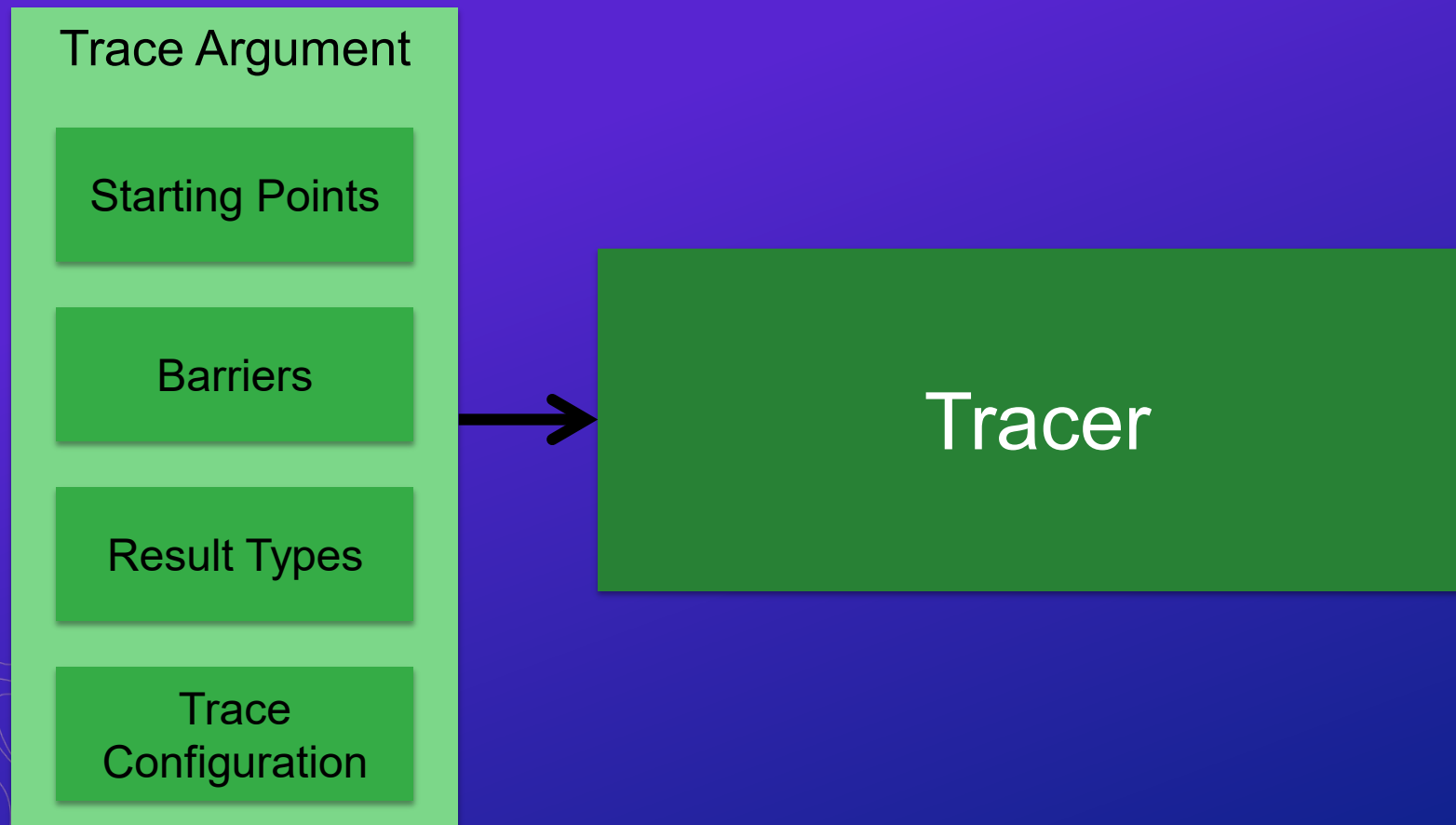


Tracer



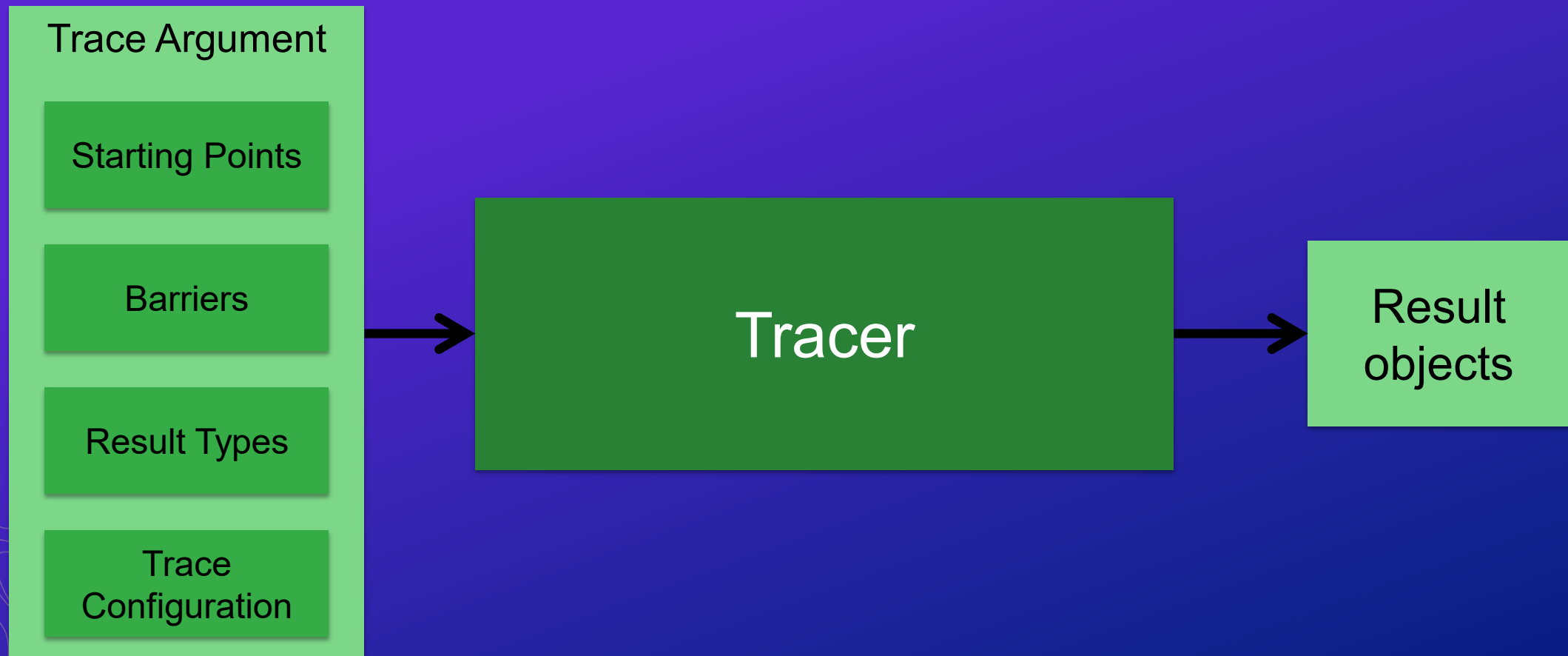
Components of a Trace

- A Trace Argument encapsulates the input parameters for a trace



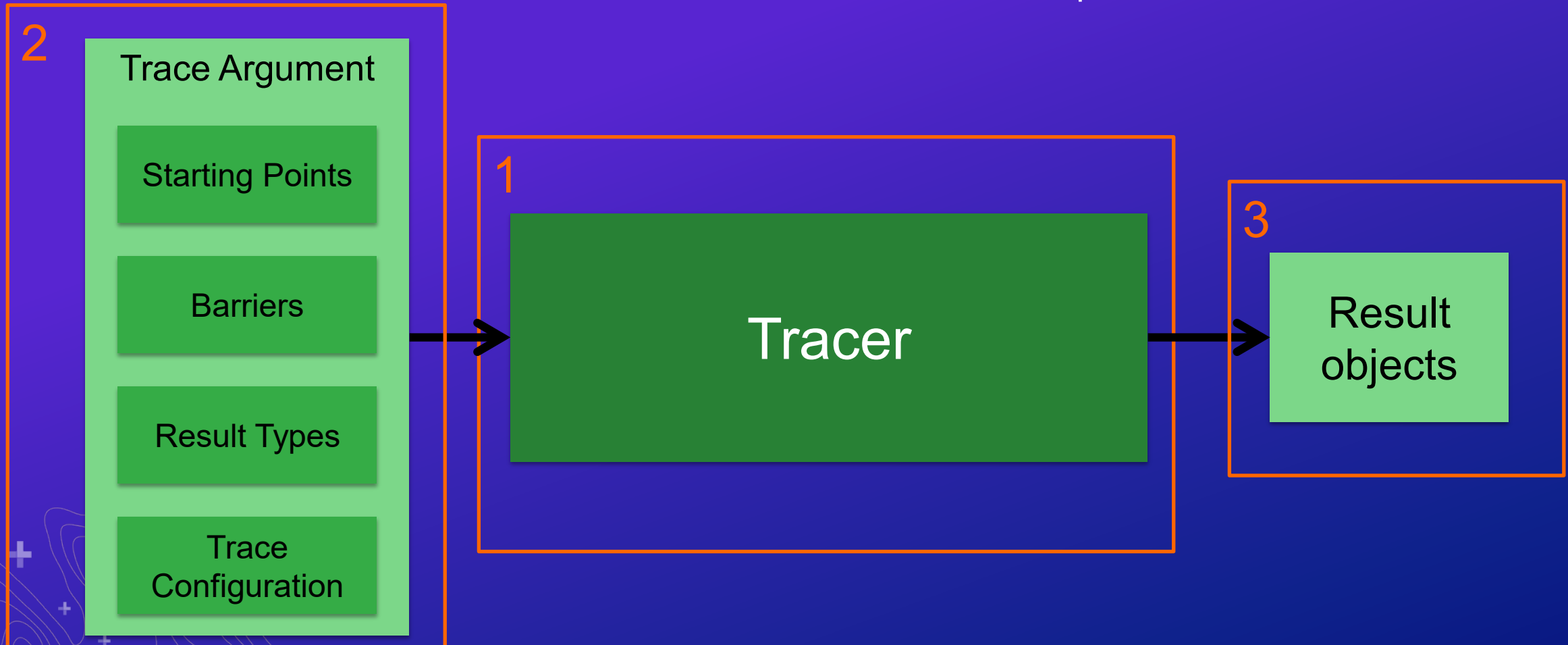
Components of a Trace

- The Tracer generates a set of Result objects as output



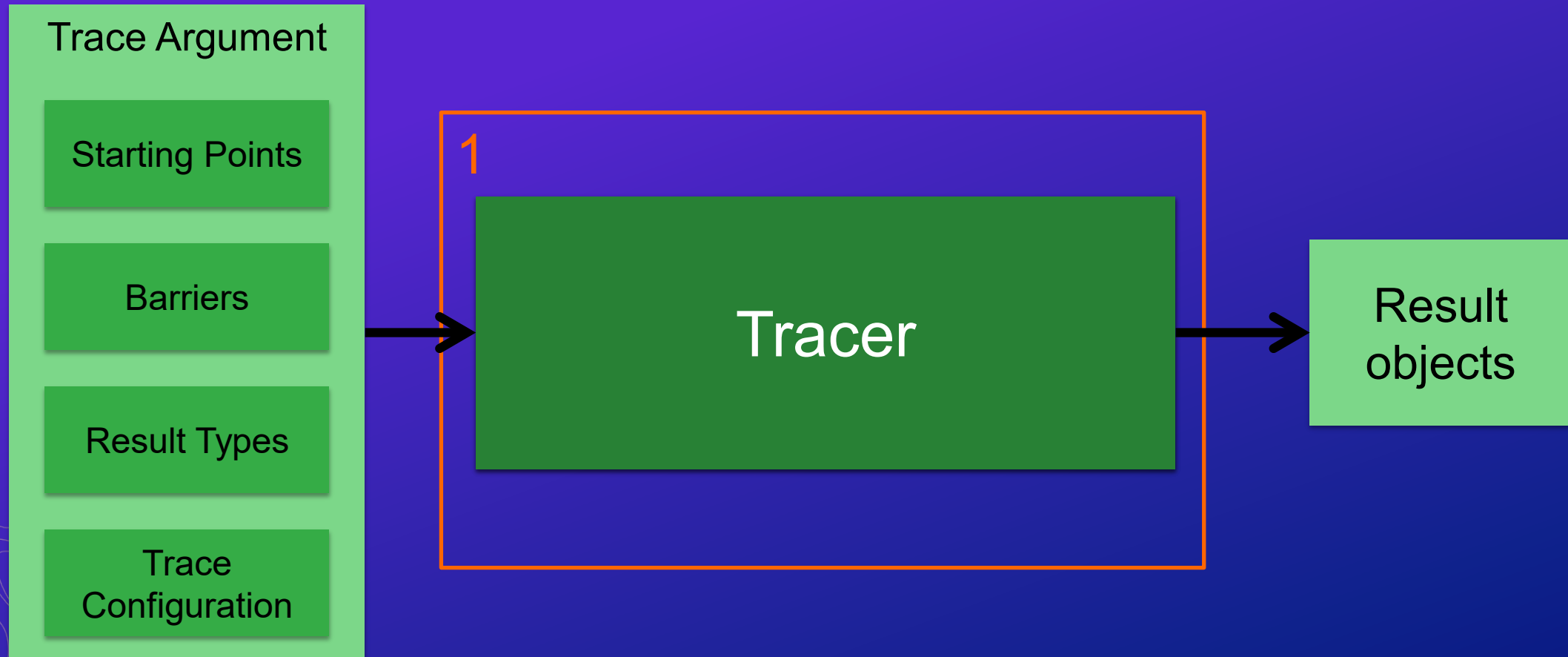
Components of a Trace

We'll cover each of these parts in more detail

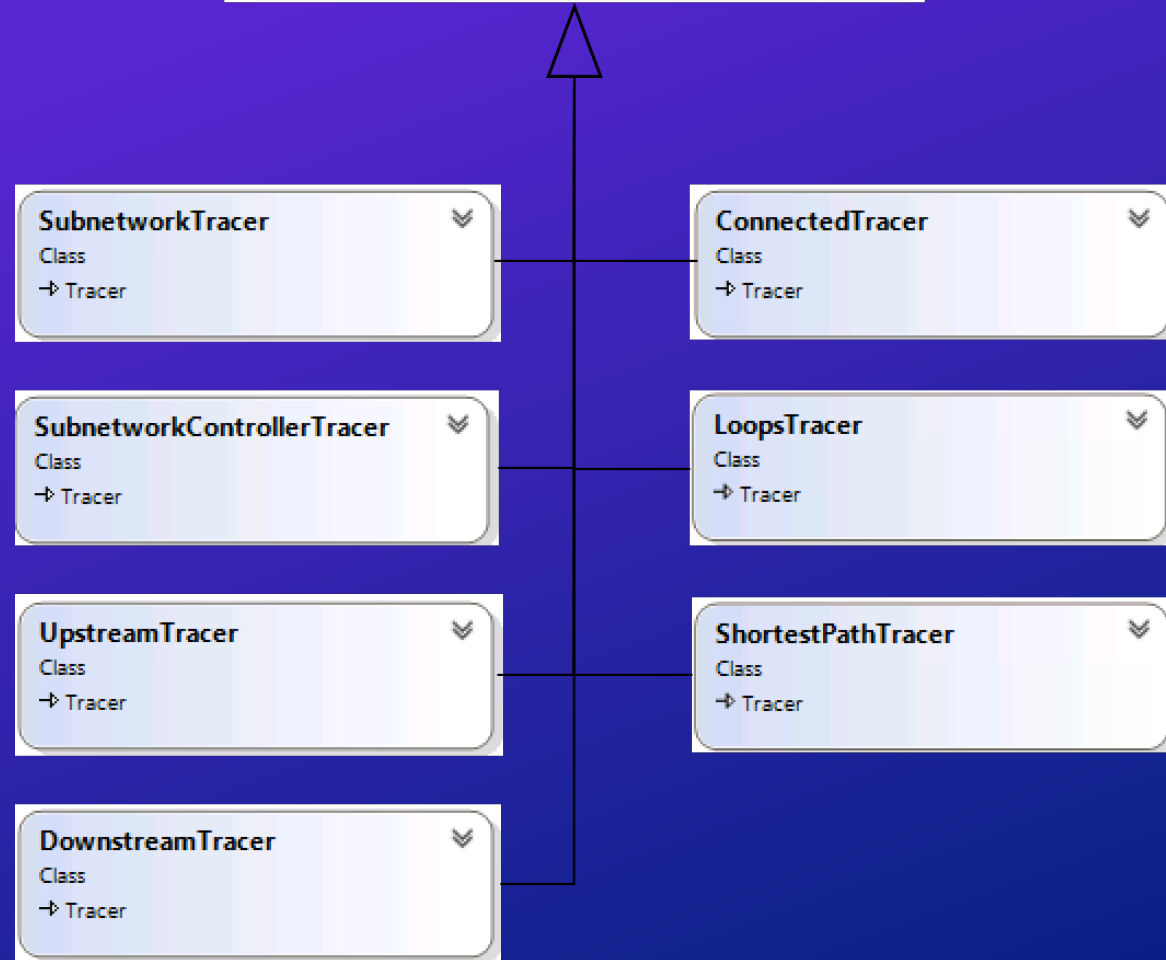
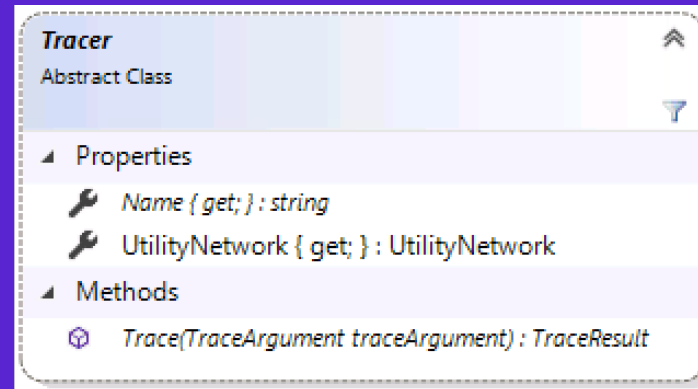


1 Tracer

- Tracers define the tracing algorithm to be used
- Tracer objects are created using `TraceManager`

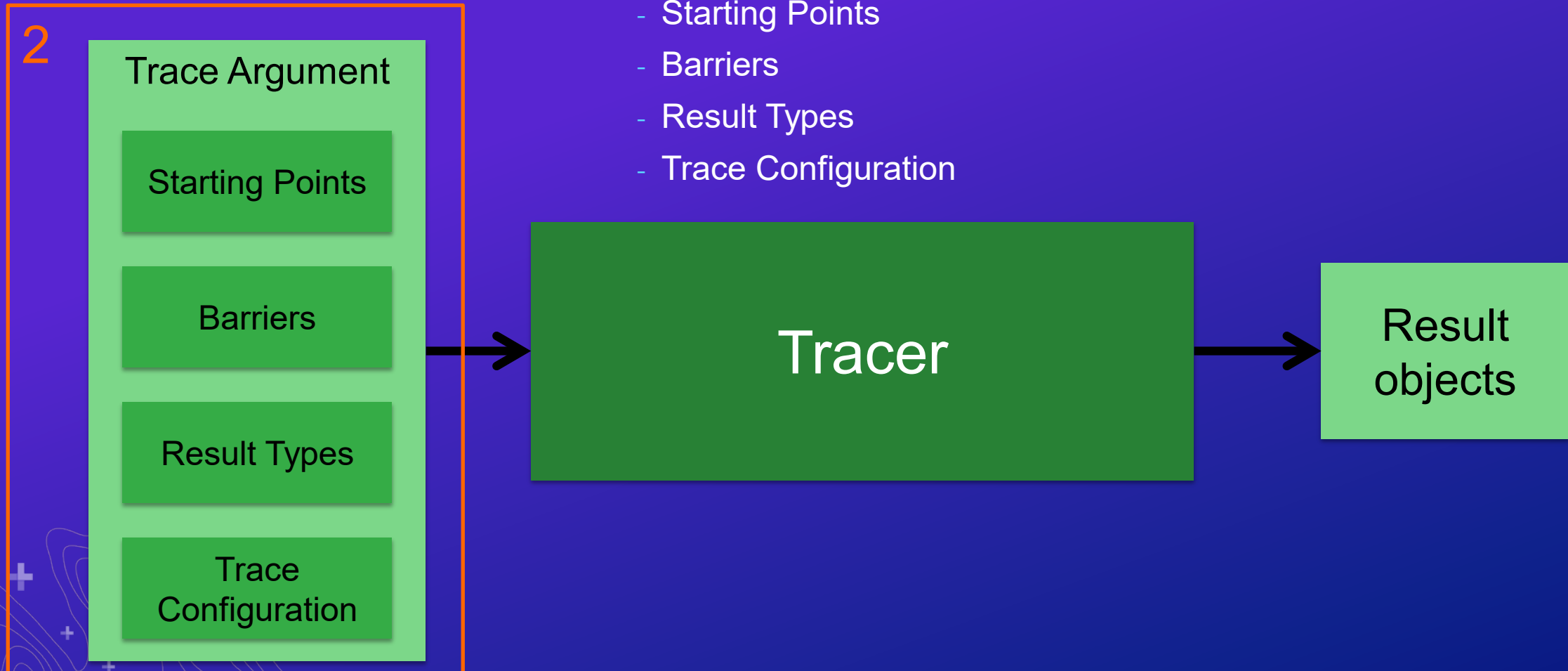


Tracer Objects



2 Trace Argument

- The `TraceArgument` class consolidates trace parameters
 - Starting Points
 - Barriers
 - Result Types
 - Trace Configuration



Trace Configuration — Basic Properties

IncludeContainers: bool

IncludeContent: bool

IncludeStructures: bool

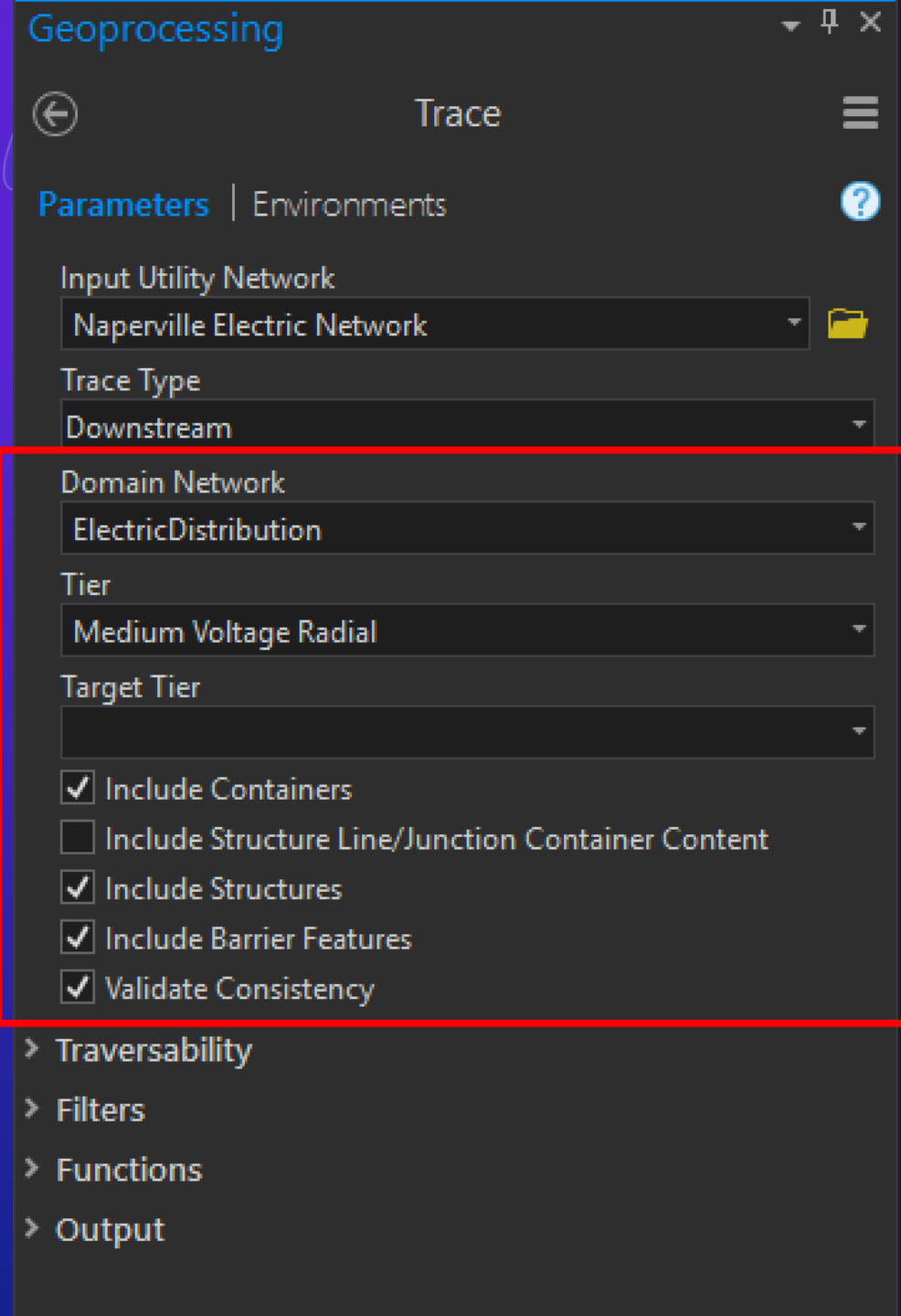
IncludeBarriersWithResults : bool

DomainNetwork : DomainNetwork

SourceTier : Tier

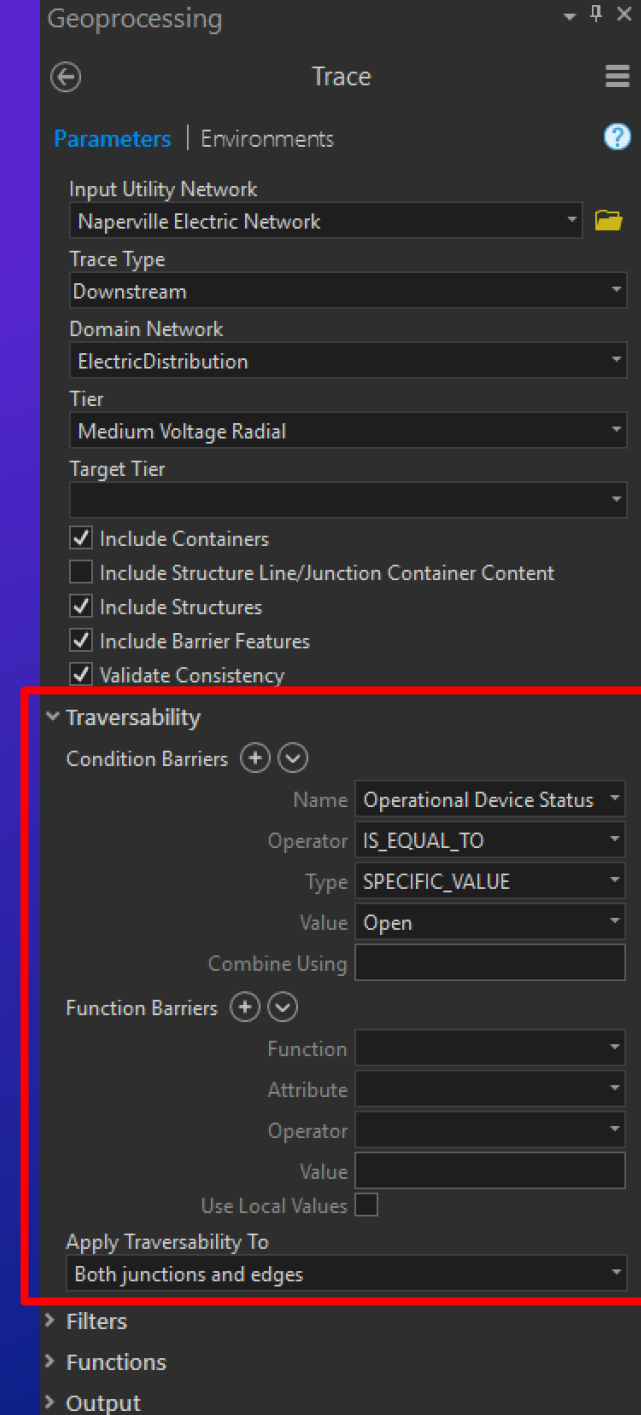
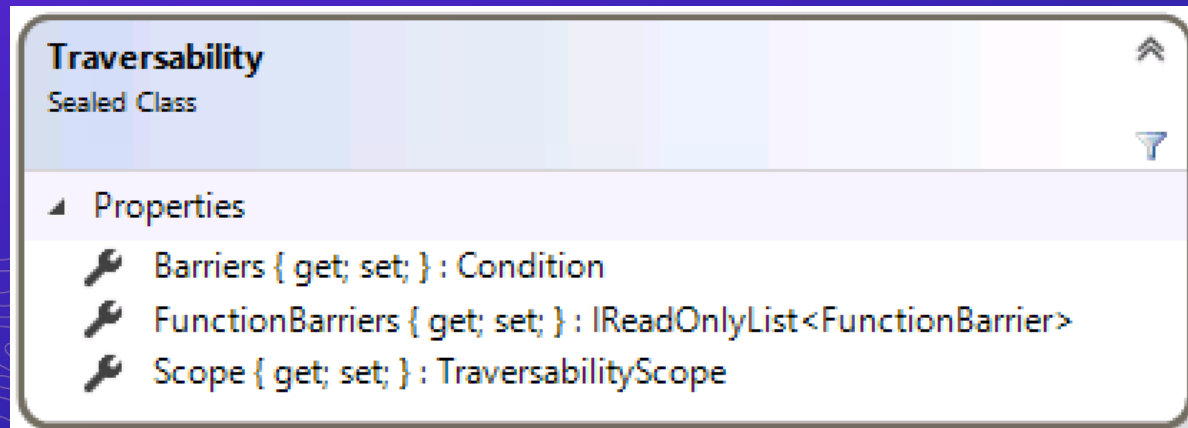
TargetTier : Tier

ValidateConsistency : bool



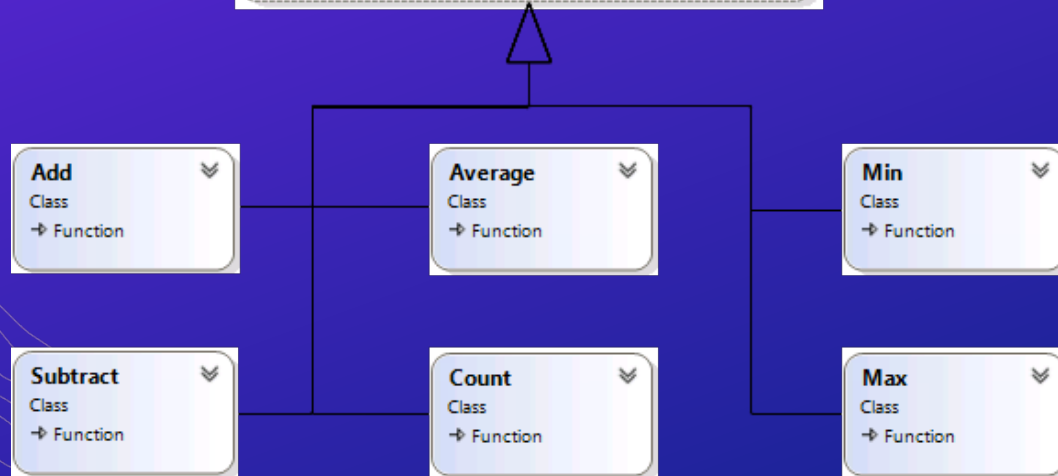
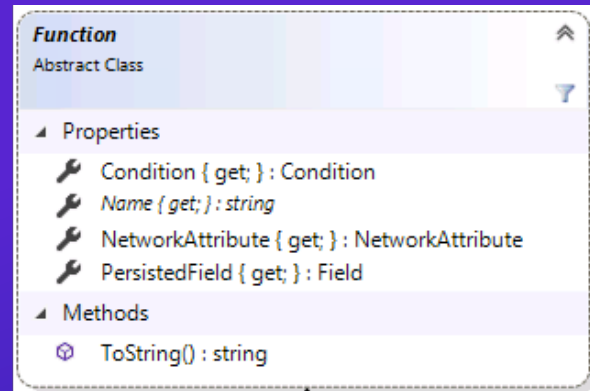
Trace Configuration — Traversability

- Traversability is based on
 - Barriers
 - Comparisons of network attributes (`NetworkAttributeComparison`), or
 - Checking for the existence of a Category (`CategoryComparison`)
 - These can be combined with boolean And and Or operations to form more complex filters
 - Function Barriers
 - Evaluation of a functional expression



Trace Configuration — Functions

- The caller can specify a collection of functions for a trace
 - These functions calculate values based on a network attribute
- At the conclusion of the trace, function results can be obtained



Geoprocessing

Trace

Parameters | Environments

Input Utility Network: Naperville Electric Network

Trace Type: Downstream

Domain Network: ElectricDistribution

Tier: Medium Voltage Radial

Target Tier: [Empty]

Include Containers

Include Structure Line/Junction Container Content

Include Structures

Include Barrier Features

Validate Consistency

> Traversability

> Filters

> **Functions**

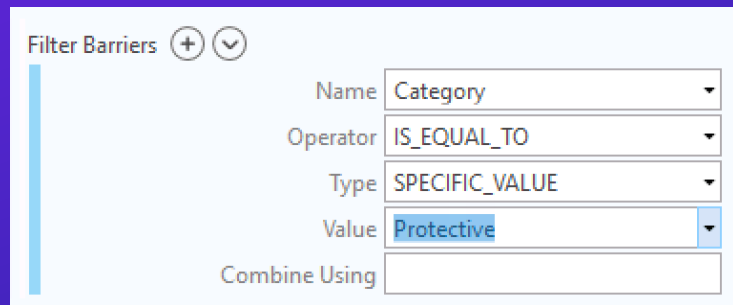
Functions (+) (-)

Function	Add
Attribute	Transformer Load
Filter Name	Phases Current
Filter Operator	INCLUDES_THE_VALUES
Filter Type	SPECIFIC_VALUE
Filter Value	A

> Output

Trace Configuration — Filters

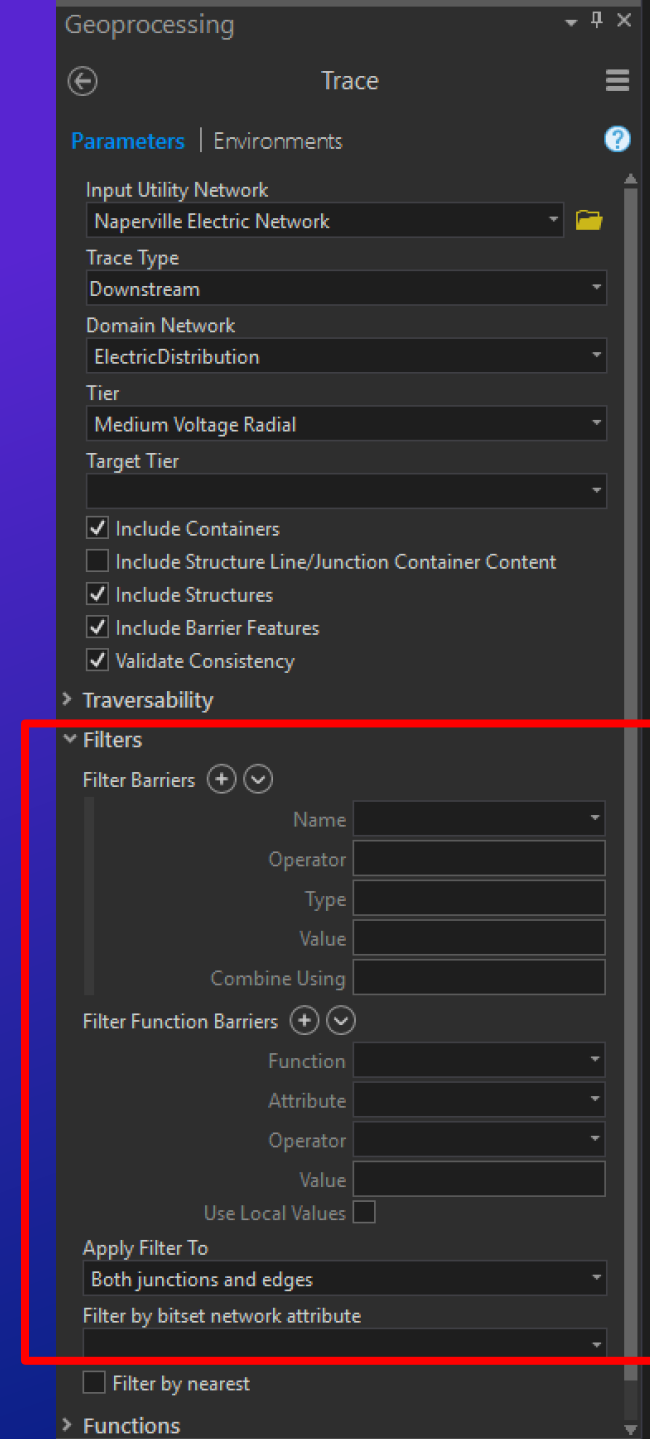
- Filters are a mechanism to stop tracing when returning results
- They do not stop traversability to the controller
- For example, returning the next upstream protective device
 - Create a filter as follows:



Filter Barriers (+) (-)

Name	Category
Operator	IS_EQUAL_TO
Type	SPECIFIC_VALUE
Value	Protective
Combine Using	

- If you tried to do this with traversability, the trace would fail because this condition would prevent finding the subnetwork source



Geoprocessing Trace

Parameters | Environments

Input Utility Network
Naperville Electric Network

Trace Type
Downstream

Domain Network
ElectricDistribution

Tier
Medium Voltage Radial

Target Tier

Include Containers
 Include Structure Line/Junction Container Content
 Include Structures
 Include Barrier Features
 Validate Consistency

> Traversability

Filters

Filter Barriers (+) (-)

Name	Category
Operator	IS_EQUAL_TO
Type	SPECIFIC_VALUE
Value	Protective
Combine Using	

Filter Function Barriers (+) (-)

Function	
Attribute	
Operator	
Value	

Use Local Values

Apply Filter To
Both junctions and edges

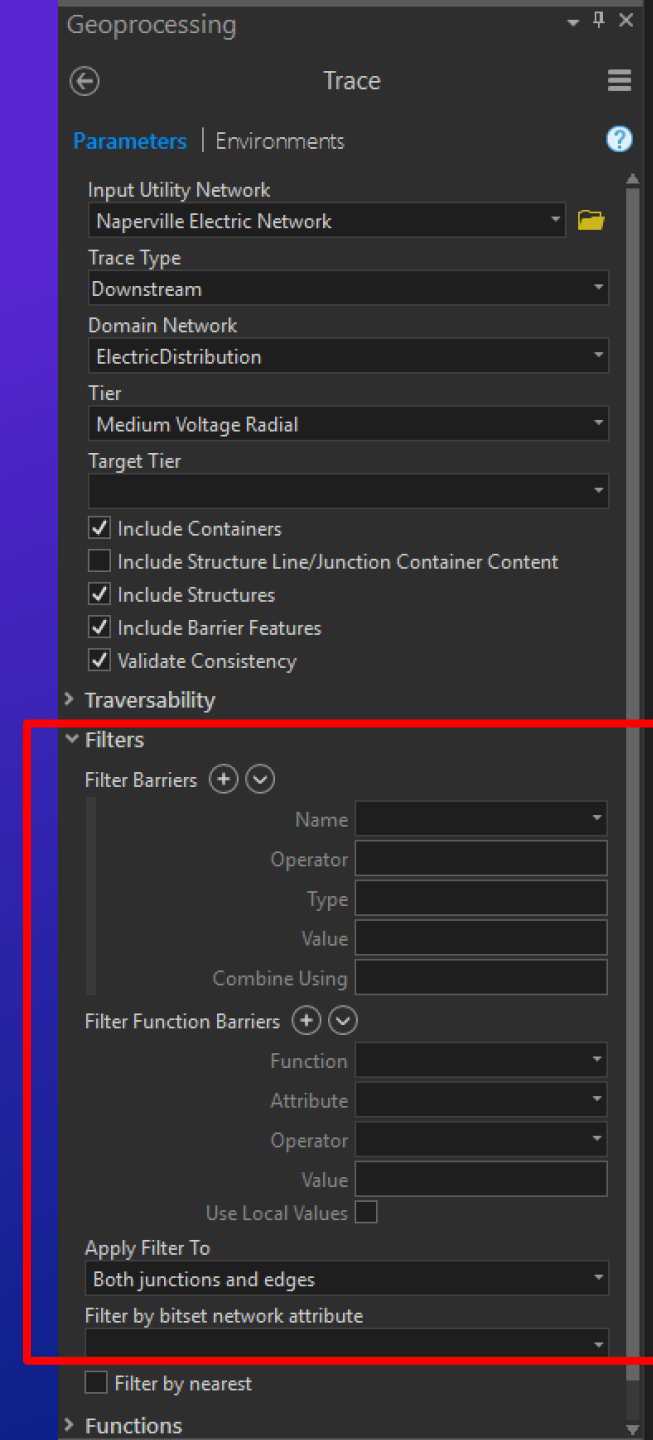
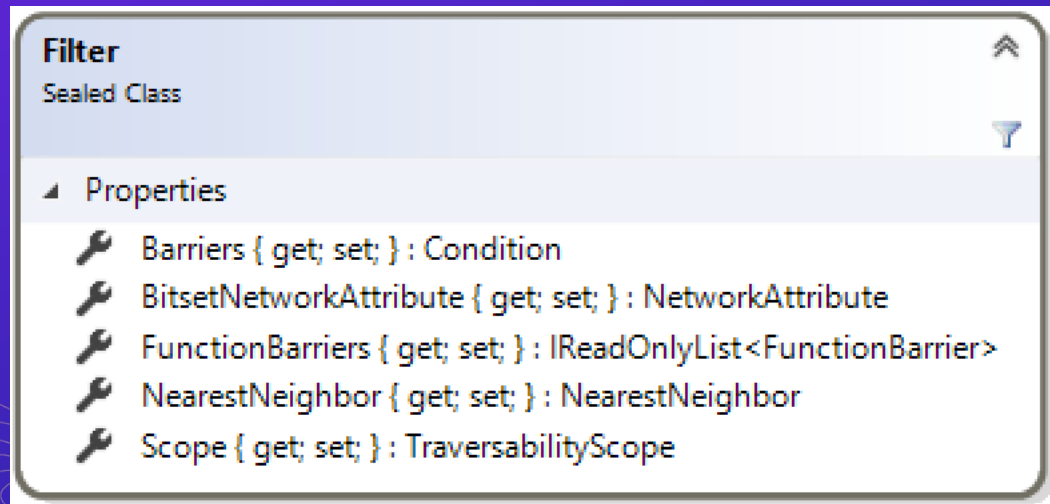
Filter by bitset network attribute

Filter by nearest

> Functions

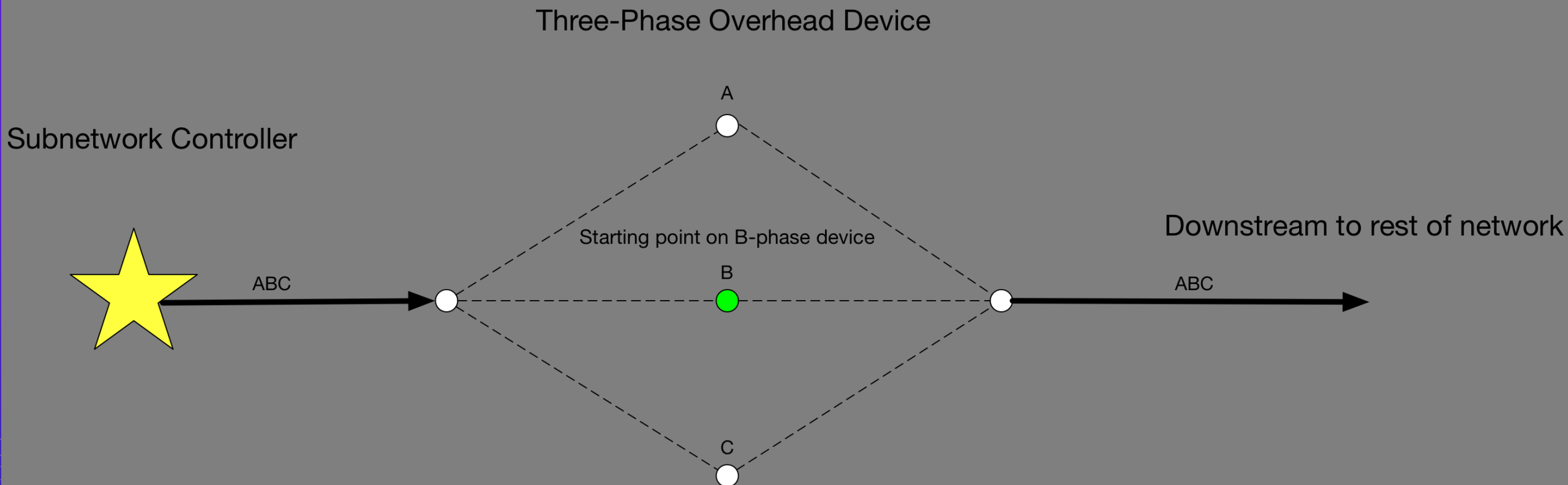
Filters

- There are a number of different ways to define filters
 - Use a Condition based on network attributes or categories
 - Use a functional expression
 - Use a bitset filter
 - Use a nearest neighbor filter



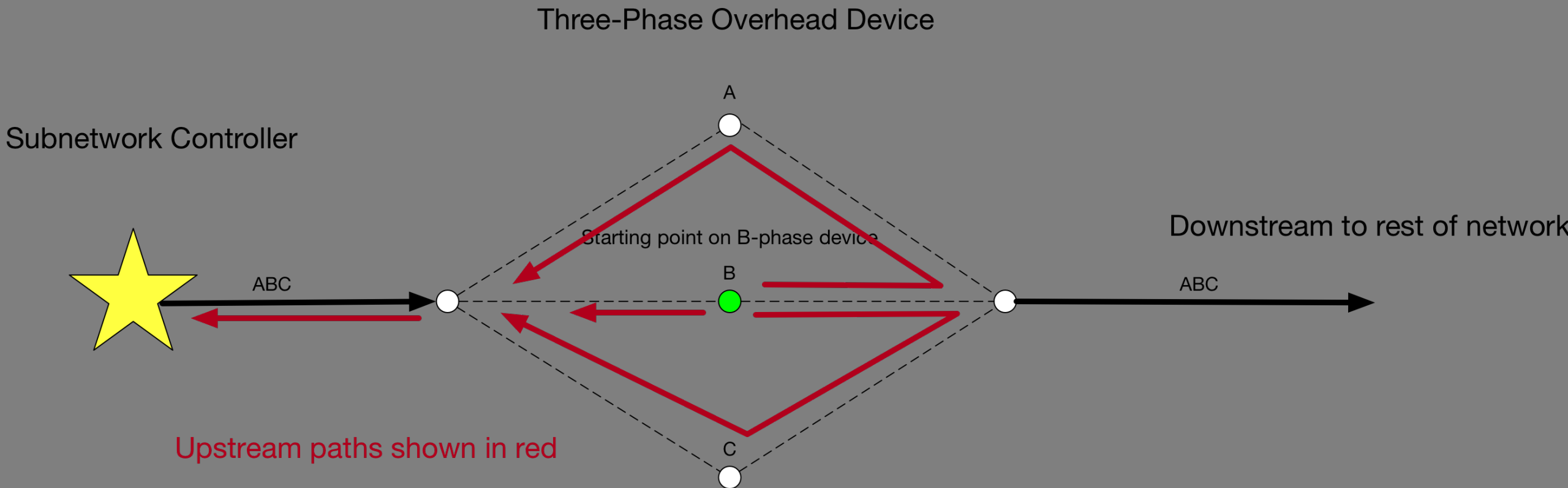
Filter by Bitset Network Attribute - 1

- There are cases where traces need to be aware that a network attribute is a bitset that controls traversability
- Consider the following network, where phase is represented as a bitset network attribute



Filter by Bitset Network Attribute - 2

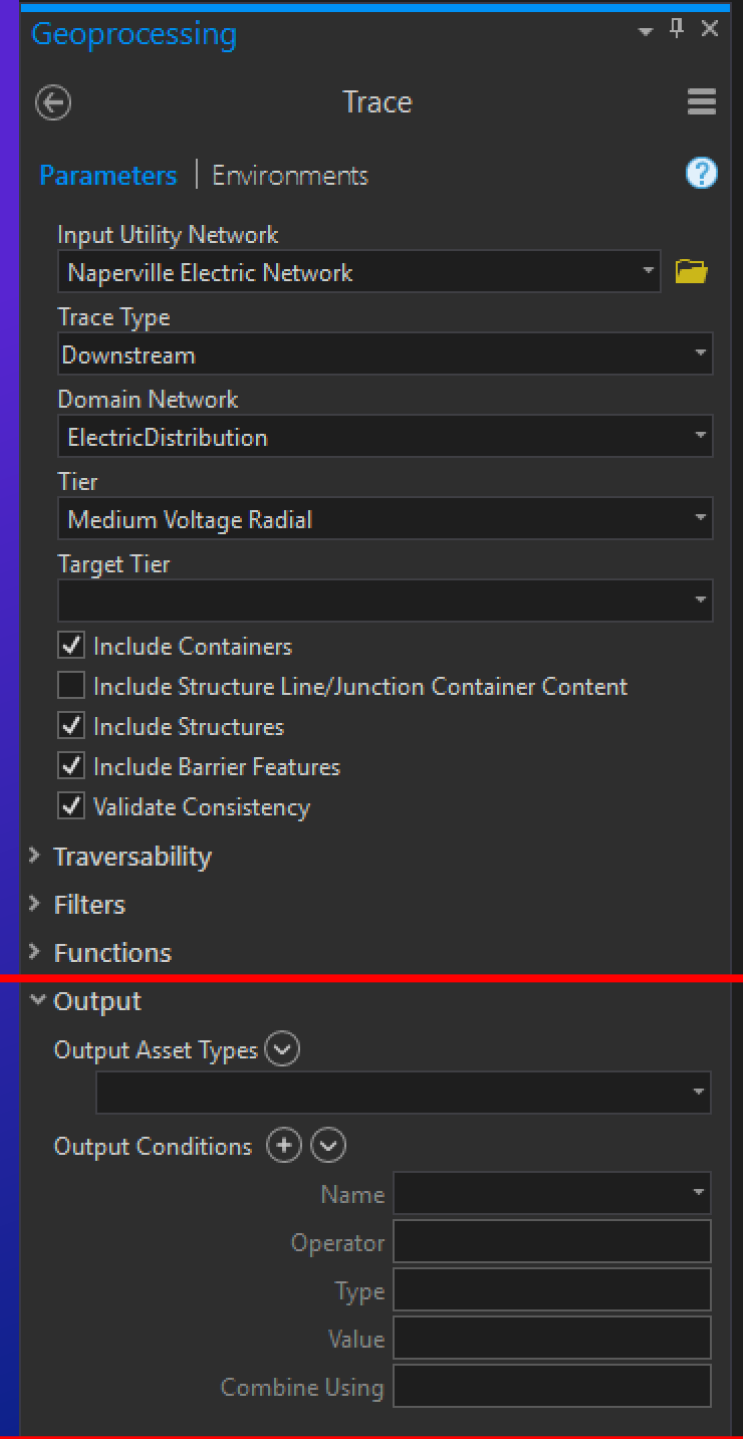
- If you do an upstream trace the results are not what you expect (electrically)



- It is applicable to downstream, upstream, and loop traces

Trace Configuration — Output

- Output filtering is applied as the final step of the tracing process
 - It takes place *after* traversability, filtering, and function calculation
- Two kinds of output filtering are provided
 - Output Condition
 - Output Asset Types



The screenshot shows the 'Trace' tool configuration window in a dark theme. The 'Output' section is highlighted with a red border. It includes a dropdown for 'Output Asset Types' and a section for 'Output Conditions' with a plus sign to add more. The 'Output Conditions' section contains fields for Name, Operator, Type, Value, and Combine Using.

Geoprocessing Trace

Parameters | Environments

Input Utility Network
Naperville Electric Network

Trace Type
Downstream

Domain Network
ElectricDistribution

Tier
Medium Voltage Radial

Target Tier

Include Containers
 Include Structure Line/Junction Container Content
 Include Structures
 Include Barrier Features
 Validate Consistency

> Traversability
> Filters
> Functions

▼ Output

Output Asset Types

Output Conditions +

Name
Operator
Type
Value
Combine Using

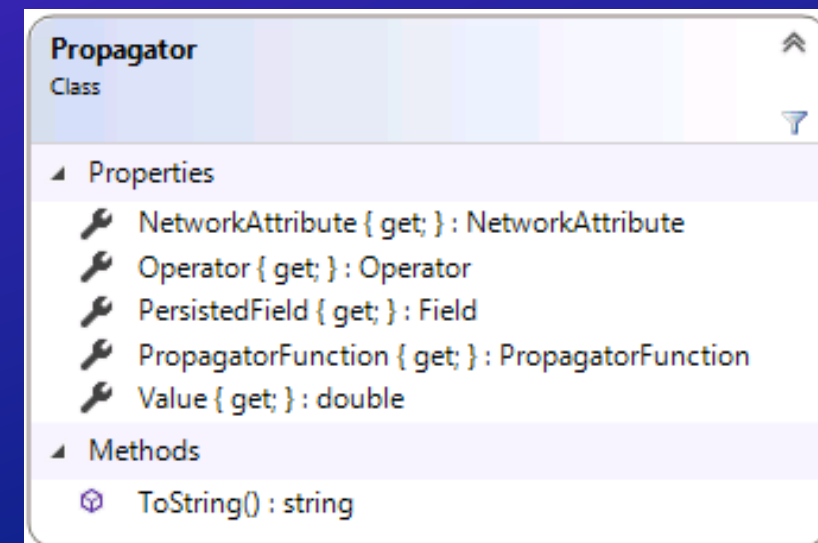
Trace Configuration — Propagators

- A propagator defines the propagation of a network attribute along a traversal, as well as provide a filter to stop traversal
- Propagators are only applicable to subnetwork-based traces (subnetwork, subnetworksource, upstream, downstream)
- The canonical example is phase propagation- open devices along the network will restrict some phases from continuing along the trace



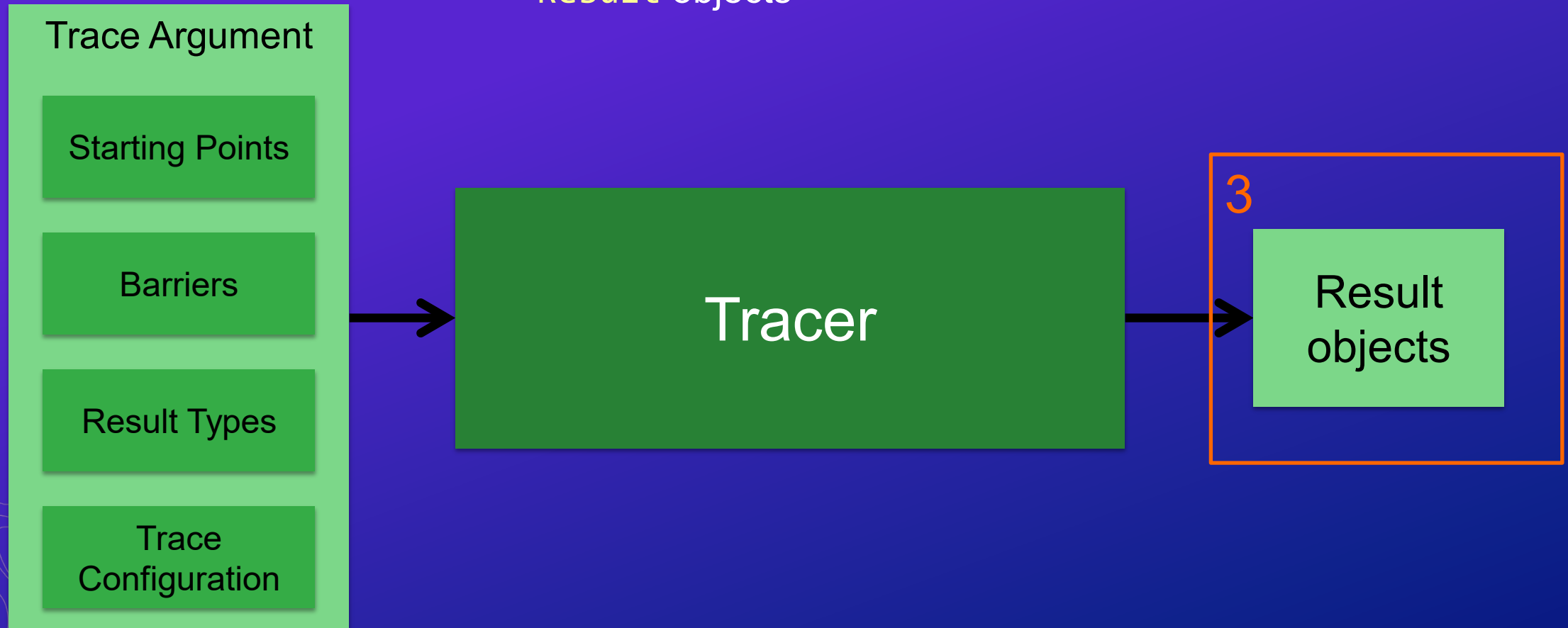
How Propagators Work

- Propagator values are computed as a pre-process step before the main trace takes place
 - Starting at each controller, the propagator uses its `PropagatorFunction` and `NetworkAttribute` to calculate a value at each element
 - This pre-process traversal covers the extent of a subnetwork
- During the trace itself, propagator filters are tested at the same time as traversal filters



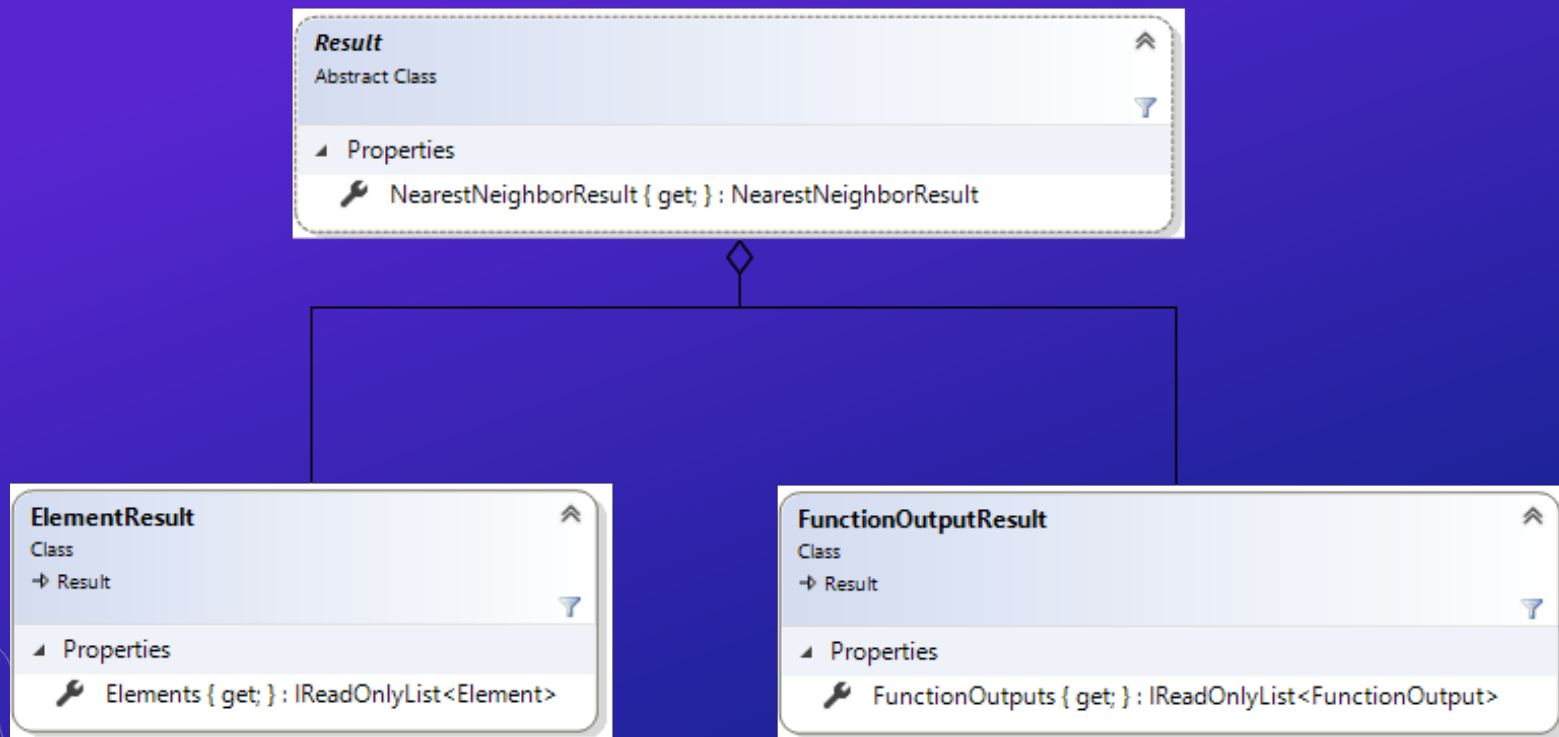
3 Trace Result

- Information is returned from a trace operation with a set of `Result` objects



Trace Results

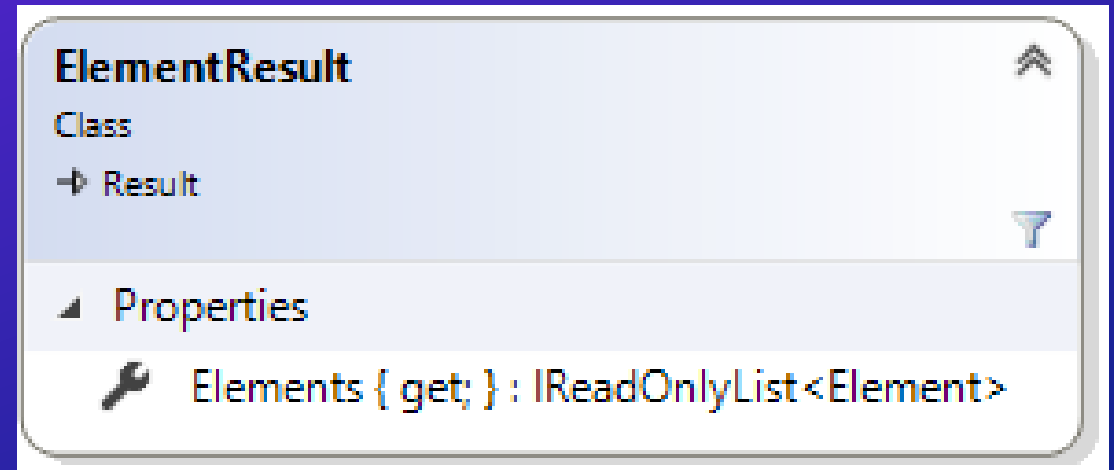
- The `Trace()` method on the `Tracer` class returns a set of `Result` objects
- One `Result` object is returned for each `ResultType` specified in `TraceArgument.ResultTypes`



Returning a List of Element Objects

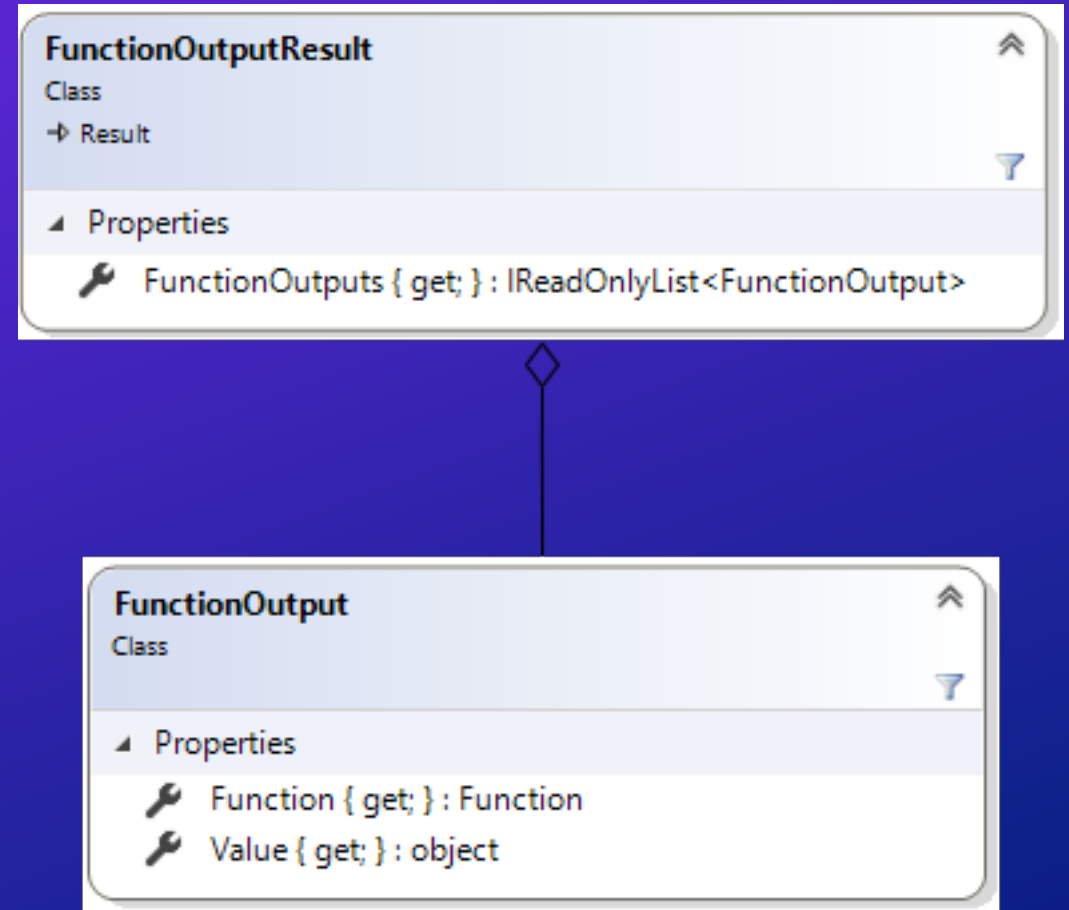
- The `ElementResult` class contains a list of `Element` objects

```
Elements : IReadOnlyList<Element>
```



Returning Function Results

- The `FunctionOutputResult` class contains a list of `FunctionOutput` objects
- The `FunctionOutput` class is a set of Function-Value pairs
 - `Function` : `Function`
 - `Value` : `object`

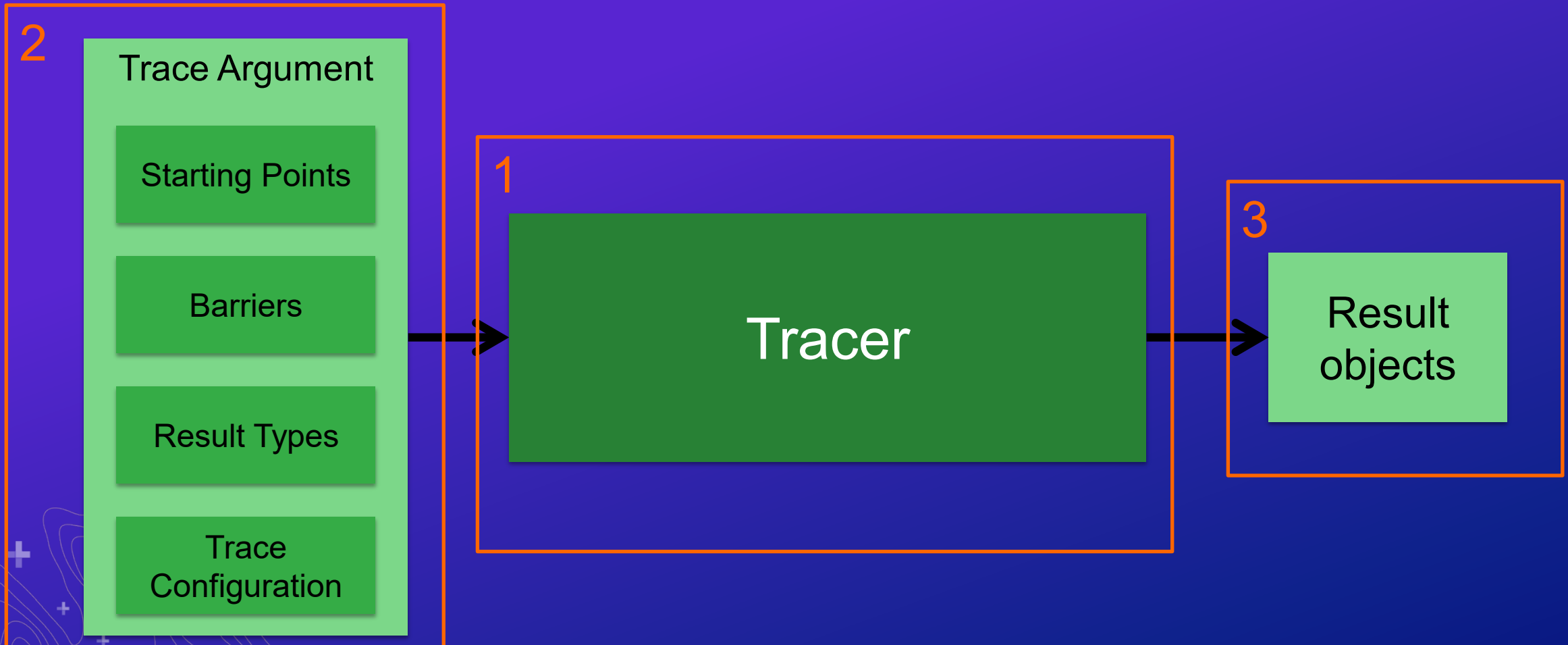


Future: Additional Result Types

- Future software versions may implement additional result types.
- Some possible result types that might be supported in the future are
 - Geometry
 - Connectivity information
 - Network Diagram
 - Propagator values per row
 - Additional network attributes
- Each result type will add a value to the `ResultType` enum and a new concrete subclass of `Result`

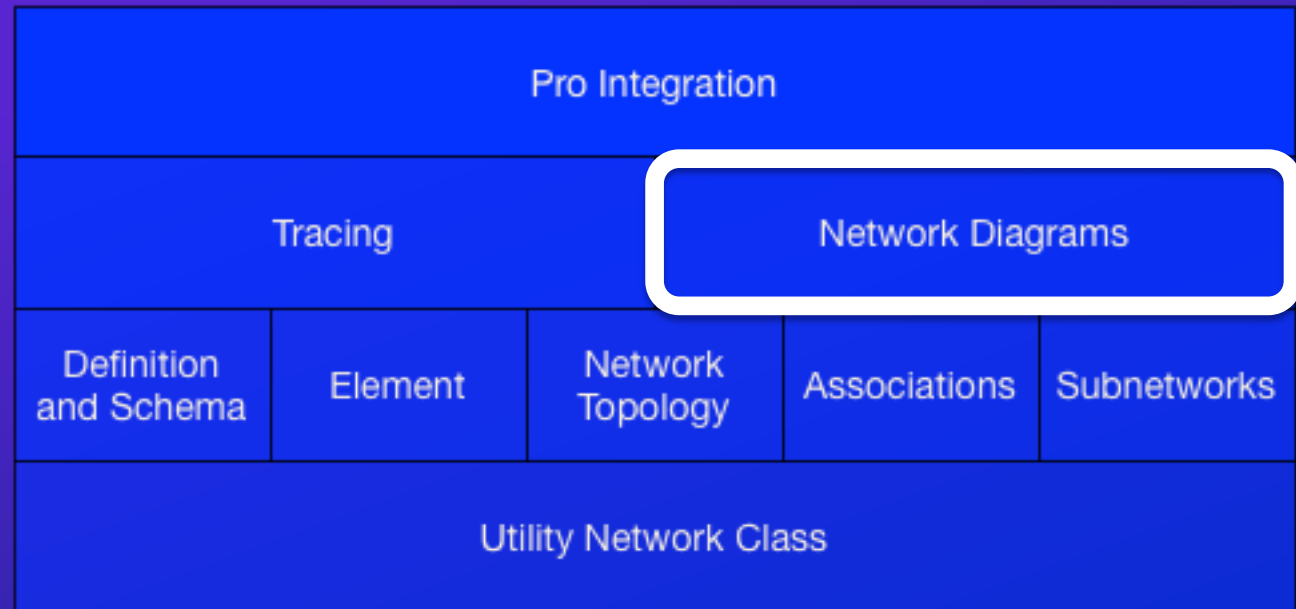


Tracing Summary



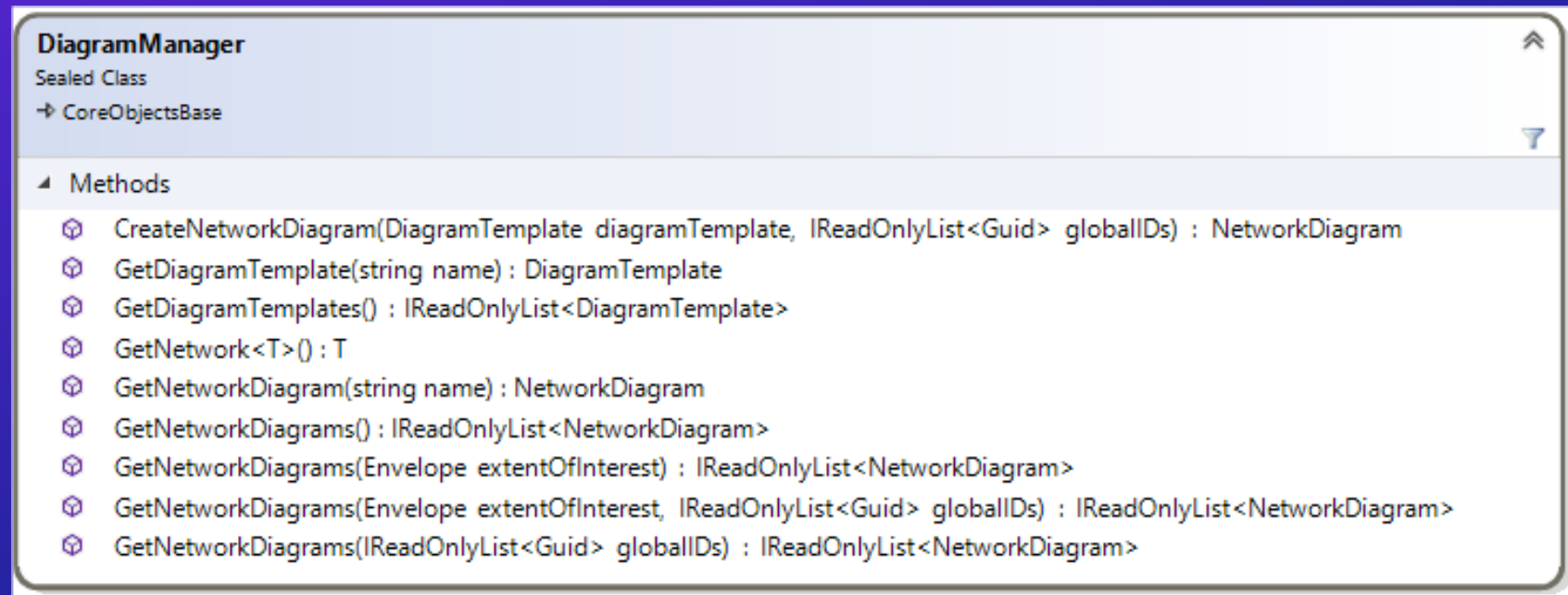
Organization of the Utility Network API

- Network Diagrams allows the developer to query and edit network diagrams



The DiagramManager Class

- The **DiagramManager** class serves as the core class in the network diagrams API
- Use the **DiagramManager** class to:
 - Create network diagrams
 - Retrieve diagram templates
 - Retrieve network diagrams
 - Retrieve the related network

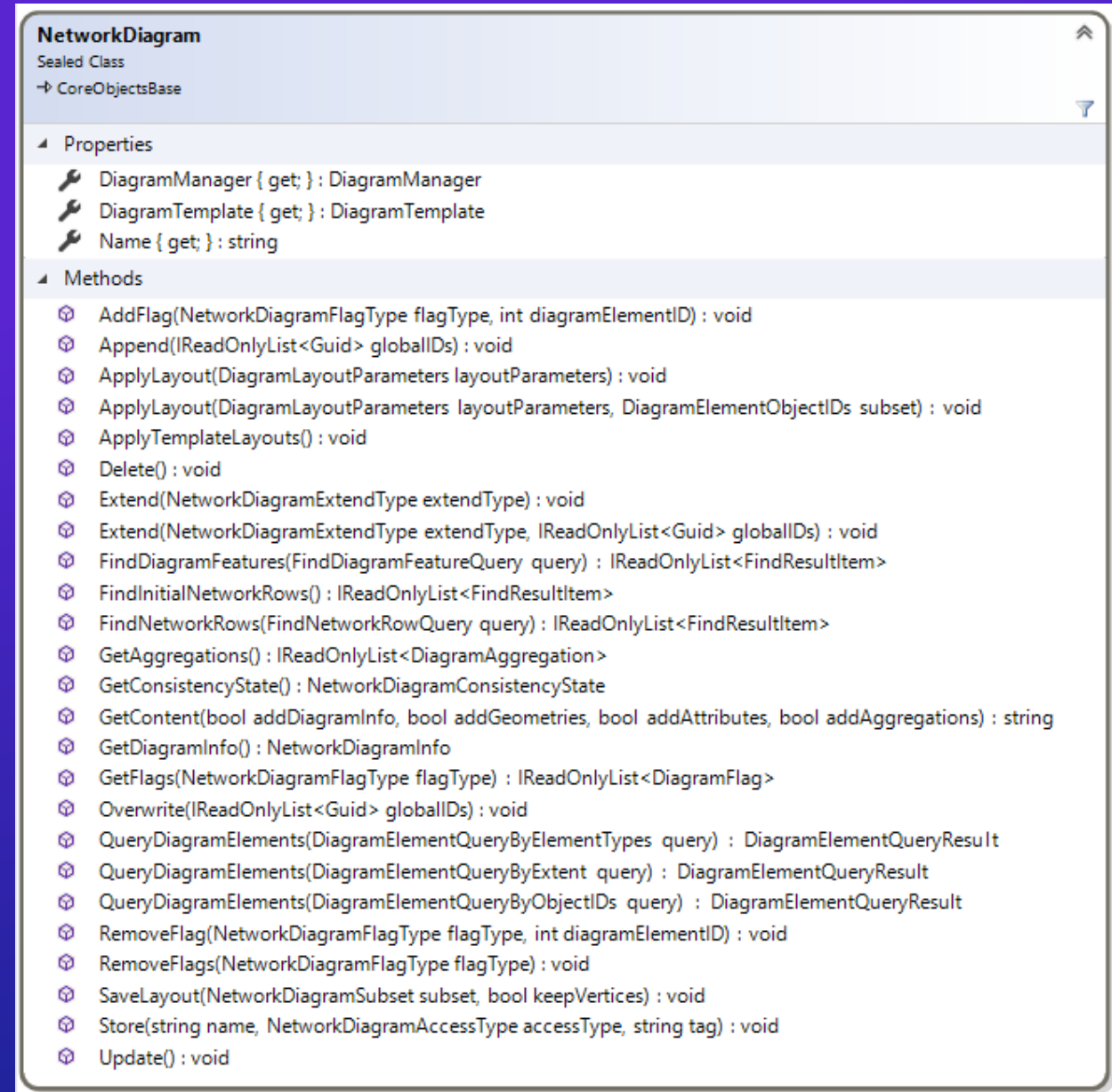


The screenshot shows the **DiagramManager** class documentation. It is a sealed class that inherits from **CoreObjectsBase**. The class has several methods listed under the 'Methods' section:

- `CreateNetworkDiagram(DiagramTemplate diagramTemplate, IReadOnlyList<Guid> globalIDs) : NetworkDiagram`
- `GetDiagramTemplate(string name) : DiagramTemplate`
- `GetDiagramTemplates() : IReadOnlyList<DiagramTemplate>`
- `GetNetwork<T>() : T`
- `GetNetworkDiagram(string name) : NetworkDiagram`
- `GetNetworkDiagrams() : IReadOnlyList<NetworkDiagram>`
- `GetNetworkDiagrams(Envelope extentOfInterest) : IReadOnlyList<NetworkDiagram>`
- `GetNetworkDiagrams(Envelope extentOfInterest, IReadOnlyList<Guid> globalIDs) : IReadOnlyList<NetworkDiagram>`
- `GetNetworkDiagrams(IReadOnlyList<Guid> globalIDs) : IReadOnlyList<NetworkDiagram>`

The NetworkDiagram Class

- This class represents a diagram generated from a portion of the utility network
- Some key routines:
 - Update
 - Append
 - Overwrite
 - Store
 - Delete
 - ApplyLayout



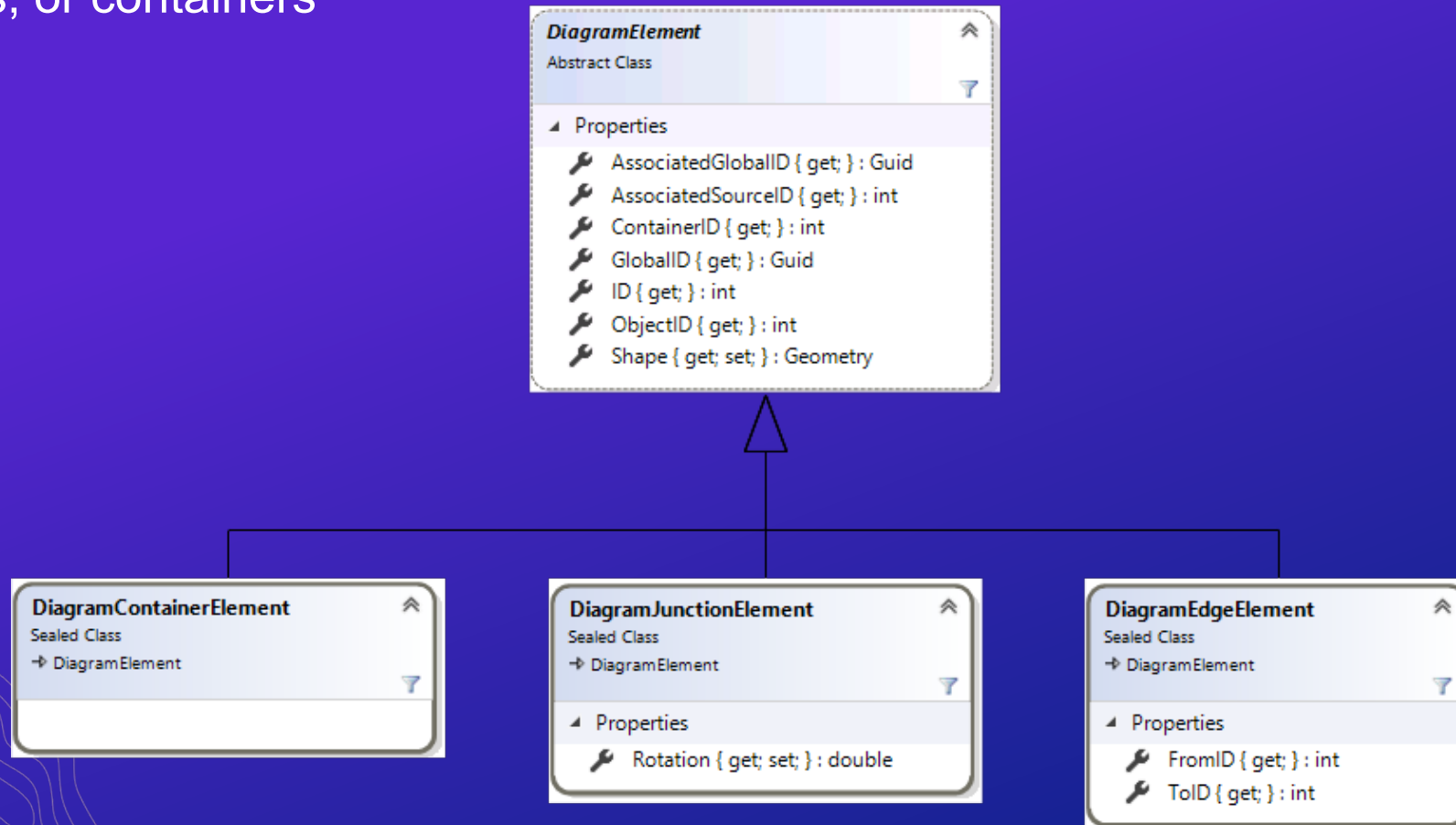
```
NetworkDiagram
Sealed Class
↳ CoreObjectsBase

Properties
DiagramManager { get; } : DiagramManager
DiagramTemplate { get; } : DiagramTemplate
Name { get; } : string

Methods
AddFlag(NetworkDiagramFlagType flagType, int diagramElementID) : void
Append(IReadOnlyList<Guid> globalIDs) : void
ApplyLayout(DiagramLayoutParameters layoutParameters) : void
ApplyLayout(DiagramLayoutParameters layoutParameters, DiagramElementObjectIDs subset) : void
ApplyTemplateLayouts() : void
Delete() : void
Extend(NetworkDiagramExtendType extendType) : void
Extend(NetworkDiagramExtendType extendType, IReadOnlyList<Guid> globalIDs) : void
FindDiagramFeatures(FindDiagramFeatureQuery query) : IReadOnlyList<FindResultItem>
FindInitialNetworkRows() : IReadOnlyList<FindResultItem>
FindNetworkRows(FindNetworkRowQuery query) : IReadOnlyList<FindResultItem>
GetAggregations() : IReadOnlyList<DiagramAggregation>
GetConsistencyState() : NetworkDiagramConsistencyState
GetContent(bool addDiagramInfo, bool addGeometries, bool addAttributes, bool addAggregations) : string
GetDiagramInfo() : NetworkDiagramInfo
GetFlags(NetworkDiagramFlagType flagType) : IReadOnlyList<DiagramFlag>
Overwrite(IReadOnlyList<Guid> globalIDs) : void
QueryDiagramElements(DiagramElementQueryByElementTypes query) : DiagramElementQueryResult
QueryDiagramElements(DiagramElementQueryByExtent query) : DiagramElementQueryResult
QueryDiagramElements(DiagramElementQueryByObjectIDs query) : DiagramElementQueryResult
RemoveFlag(NetworkDiagramFlagType flagType, int diagramElementID) : void
RemoveFlags(NetworkDiagramFlagType flagType) : void
SaveLayout(NetworkDiagramSubset subset, bool keepVertices) : void
Store(string name, NetworkDiagramAccessType accessType, string tag) : void
Update() : void
```

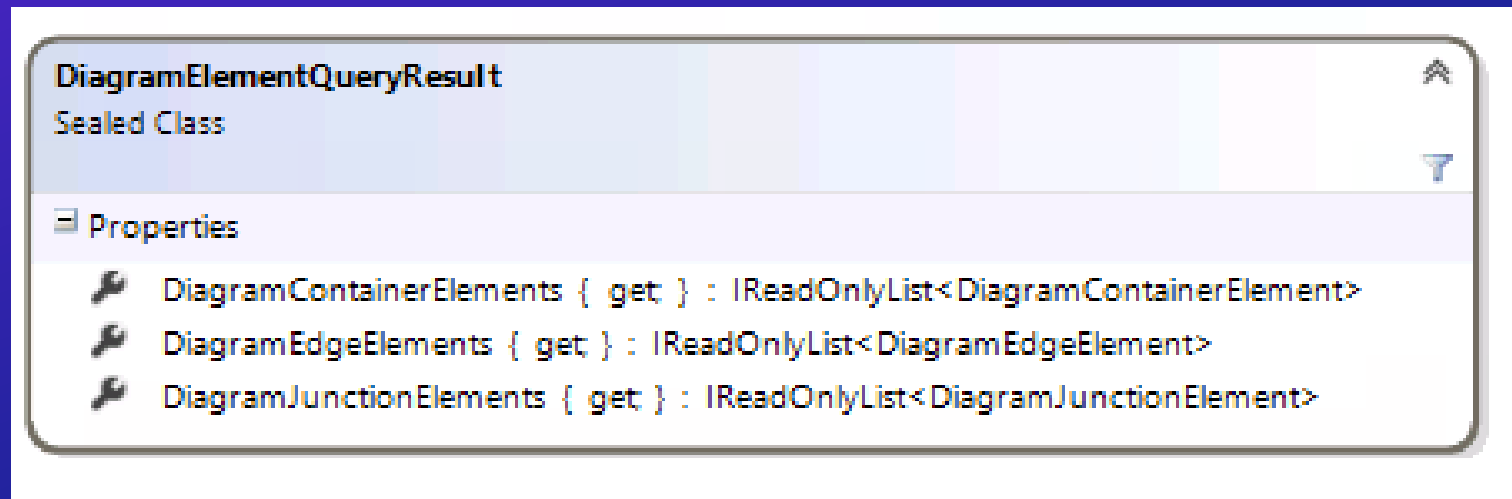
Diagram Element Classes

- A network diagram consists of a set of diagram elements which are either junctions, edges, or containers



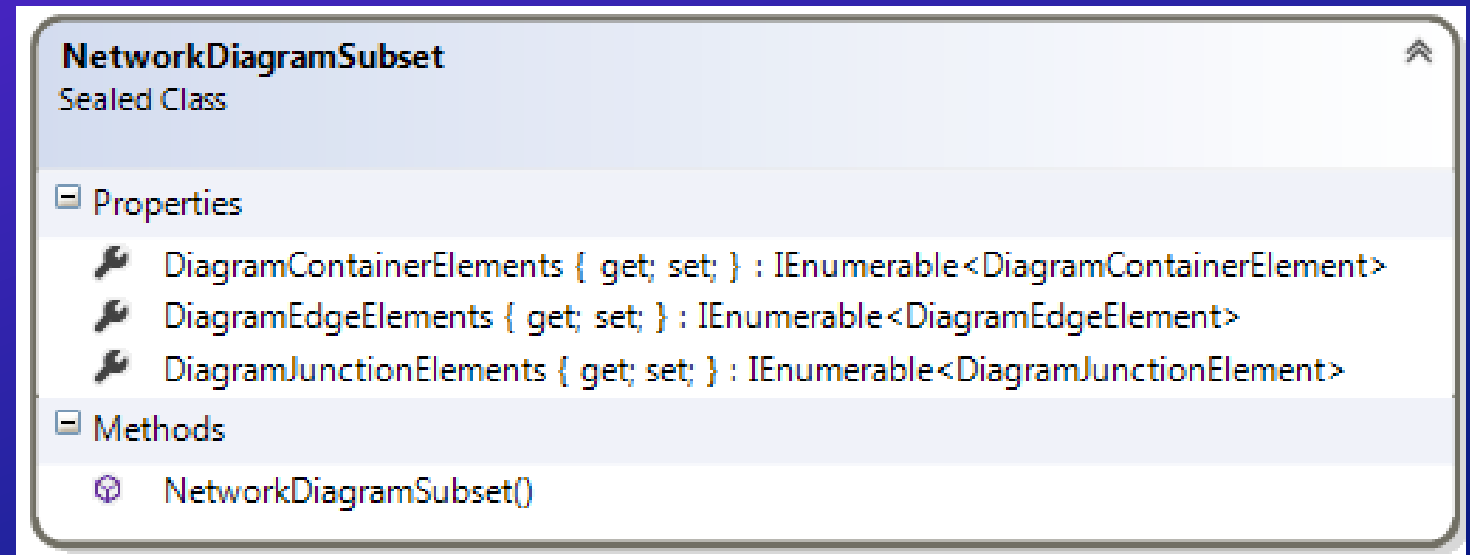
Retrieving Diagram Elements

- By Type (edge, junction, container)
 - `DiagramElementQueryByElementTypes` class
- By Extent
 - `DiagramElementQueryByExtent` class
- By set of ObjectIDs
 - `DiagramElementQueryByObjectIDs` class



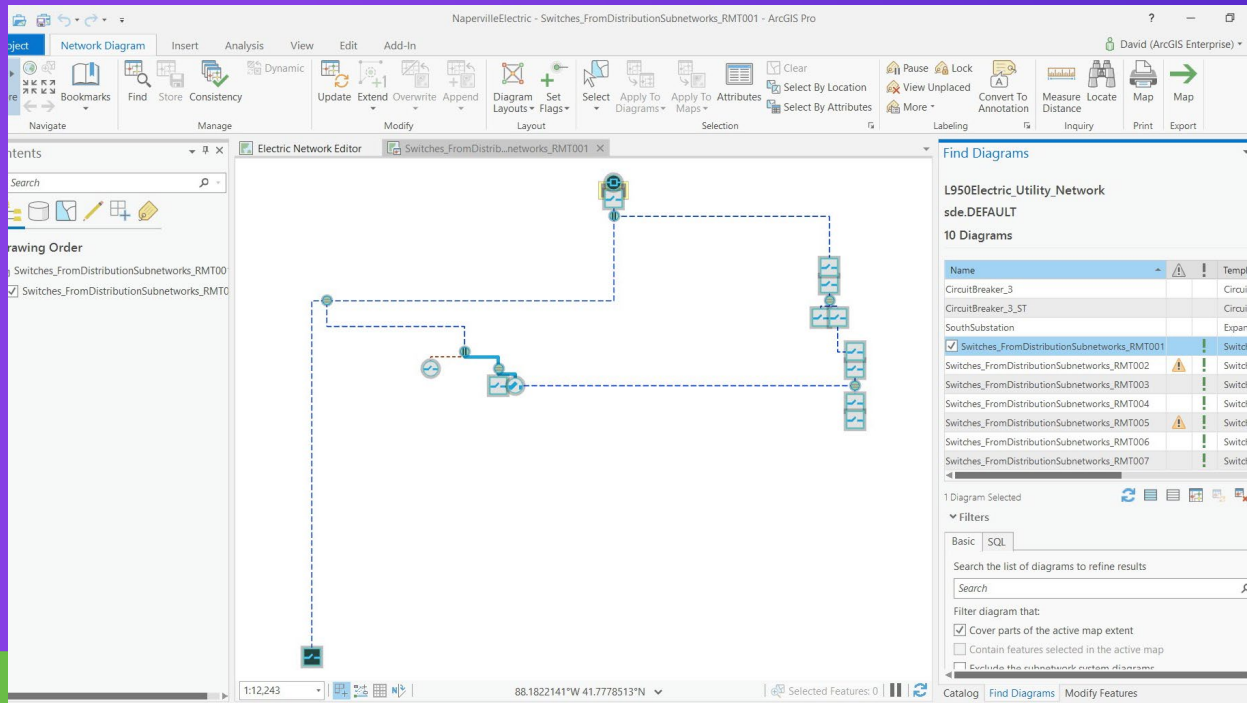
Custom Layouts

1. Query for set of diagram elements
2. Change the shape of those elements
3. Add the diagram elements to a `NetworkDiagramSubset` object
4. Pass that as an argument to `NetworkDiagram.SaveLayout()`



The screenshot shows the definition of the `NetworkDiagramSubset` class in a code editor. The class is a sealed class and contains three properties and one method.

```
NetworkDiagramSubset  
Sealed Class  
  
[-] Properties  
    DiagramContainerElements { get; set; } : IEnumerable<DiagramContainerElement>  
    DiagramEdgeElements { get; set; } : IEnumerable<DiagramEdgeElement>  
    DiagramJunctionElements { get; set; } : IEnumerable<DiagramJunctionElement>  
  
[-] Methods  
    NetworkDiagramSubset()
```

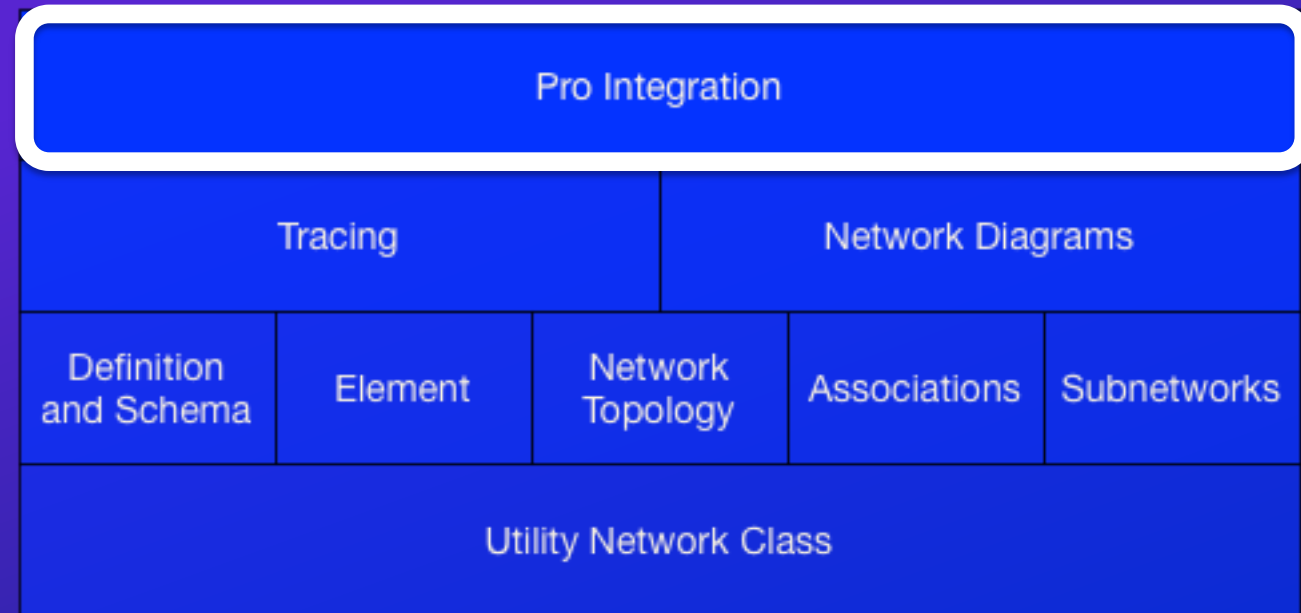


Network Diagrams

David Crawford

Organization of the Utility Network API

- Finally, Pro Integration describes how the utility network API integrates with other parts of the Pro SDK

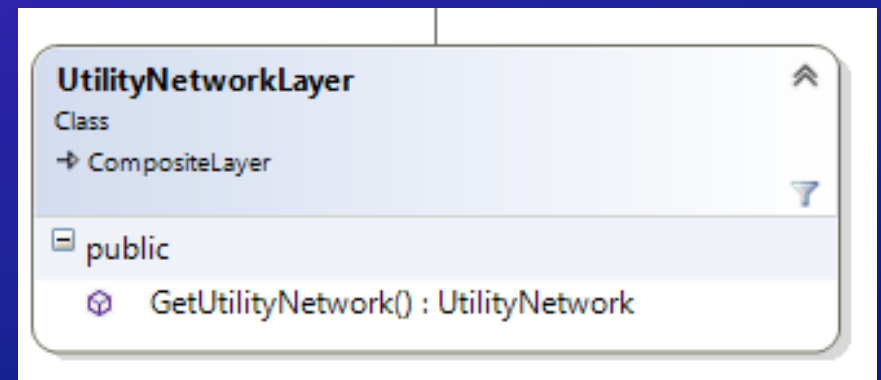


The UtilityNetworkLayer Class

- The `UtilityNetworkLayer` class represents a utility network layer in a map
- The `UtilityNetworkLayer` class inherits from `CompositeLayer`, which can be used to iterate through the dirty area and error sublayers

`GetUtilityNetwork() : UtilityNetwork`

- Returns the `UtilityNetwork` class pointed at by this layer
- Used with ArcGIS Pro add-ins to obtain the underlying geodatabase object from the selected layer



Ribbon Integration

- When you are writing a button or tool to appear on the ribbon, the config.daml file is used to configure when the object is enabled
 - [As described by the Pro SDK documentation](#)
- We provide a new utility network condition that enables items if a utility network layer is in the active map

```
condition="esri_mapping_utilityNetworkCondition"
```

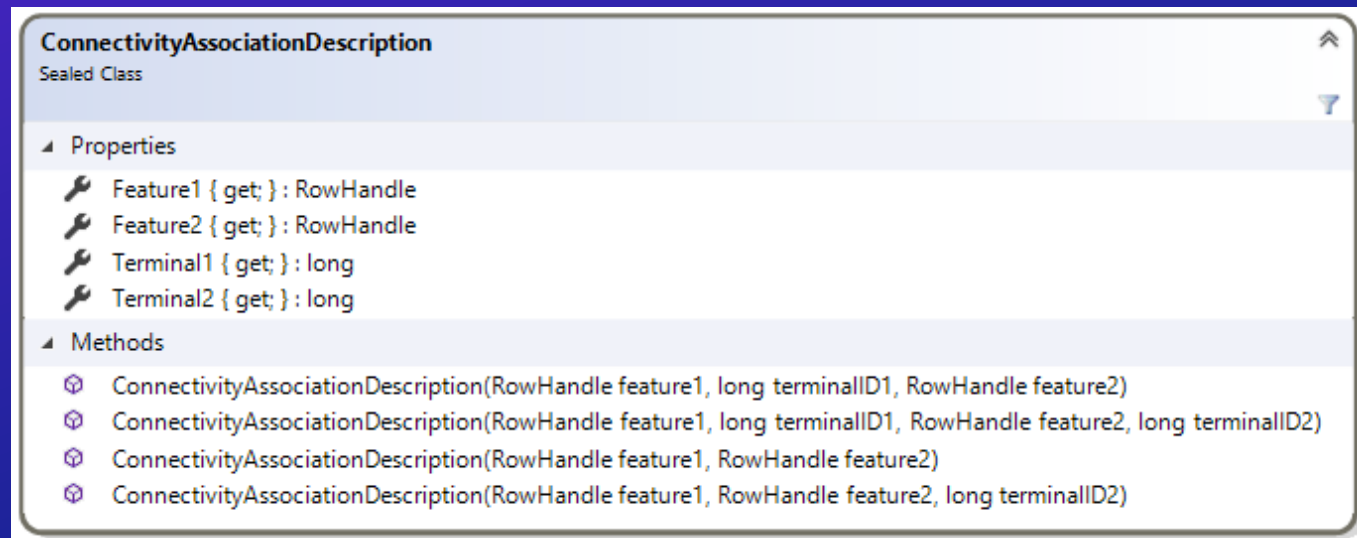
Editor Integration

- The preferred technique for editing is to use the high level `EditOperation` class
 - This class includes a number of high-level editing routines like `Create()`, `Delete()`, `Move()`
 - These routines queue up edits
 - `EditOperation.Execute()` fires off the actual edits
- The `EditOperation` was extended in 2.1 to support utility network edits

Read the [Editor Conceptual Documentation](#) for more information

Creating and Deleting Associations

- New set of AssociationDescription classes created
 - ConnectivityAssociationDescription
 - ContainmentAssociationDescription
 - StructuralAttachmentAssociationDescription
- New Create() and Delete() overloads created on EditOperation class



The screenshot displays the class definition for `ConnectivityAssociationDescription`, which is a sealed class. It is organized into two main sections: Properties and Methods.

Properties:

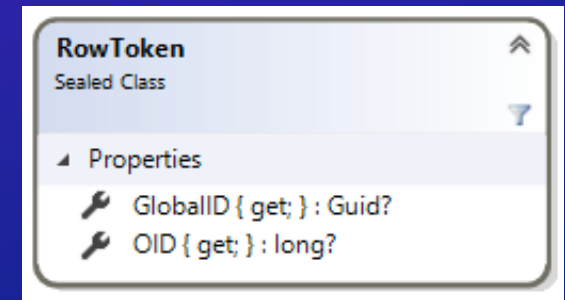
- `Feature1 { get; } : RowHandle`
- `Feature2 { get; } : RowHandle`
- `Terminal1 { get; } : long`
- `Terminal2 { get; } : long`

Methods:

- `ConnectivityAssociationDescription(RowHandle feature1, long terminalID1, RowHandle feature2)`
- `ConnectivityAssociationDescription(RowHandle feature1, long terminalID1, RowHandle feature2, long terminalID2)`
- `ConnectivityAssociationDescription(RowHandle feature1, RowHandle feature2)`
- `ConnectivityAssociationDescription(RowHandle feature1, RowHandle feature2, long terminalID2)`

Creating Features and Associations in the Same Edit Operation

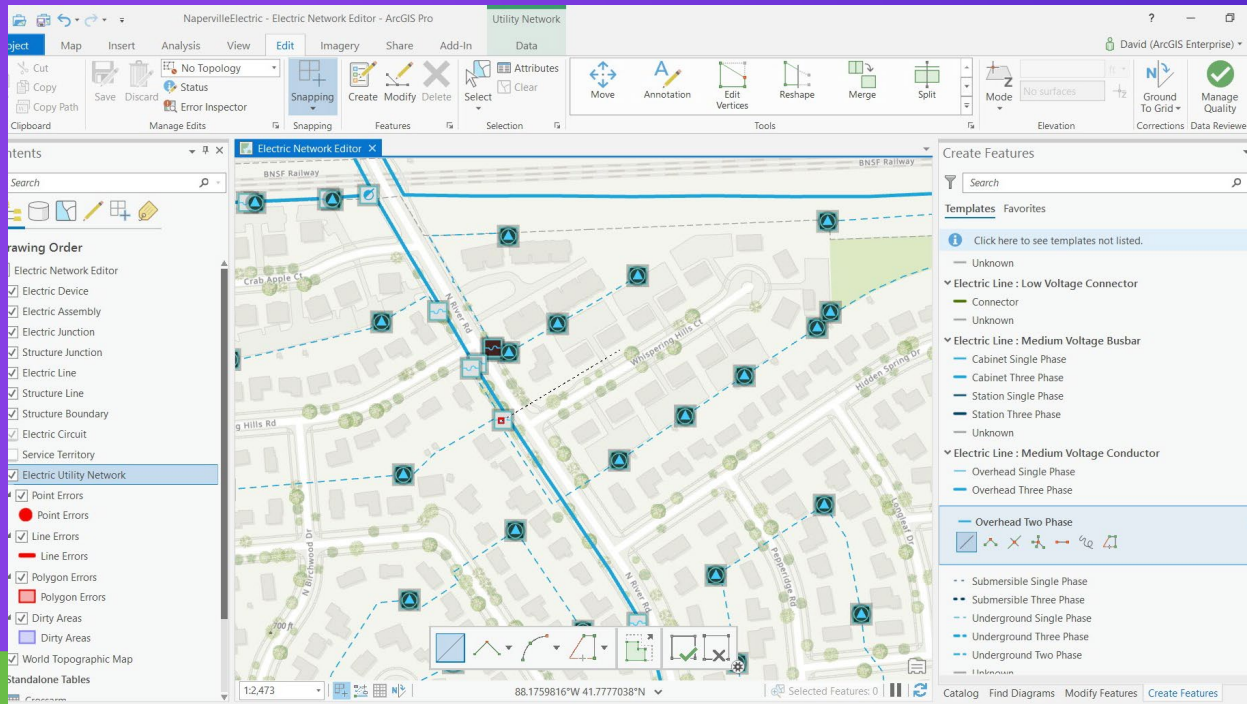
- How do you specify an association between two features when the features haven't been created yet?
 - You don't have an ObjectID or GlobalID
- New `CreateEx()` methods on `EditOperation` return a `RowToken`
 - This token can be passed into `AssociationDescription` classes



Editing with low-level Geodatabase Routines

- We also provide low-level geodatabase objects for editing
- This is the only way to edit in a stand-alone application
- Not recommended in an ArcGIS Pro Add-in
 - The Map is not refreshed
 - The Undo/Redo stack is not modified



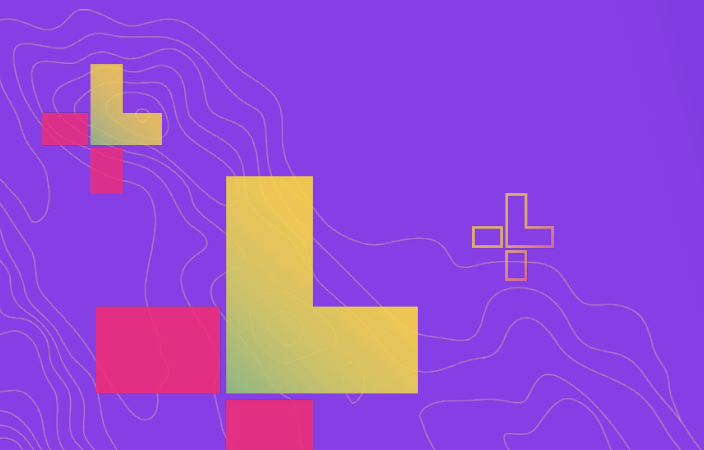


Editing

David Crawford

What's New

ArcGIS Pro 2.4 and 2.5



New Since DevSummit 2019

- Isolation tracing with the IsolationTracer class 2.4
- And and Or operator precedence 2.5
- New TraceConfiguration properties 2.4
 - IgnoreBarriersAtStartingPoints
 - IncludeIsolatedFeatures
- Support attribute substitution with propagation 2.4
- Support subnetwork retrieval by name 2.4
- Enhancements to Attribute Rule APIs (Geodatabase) 2.4
 - Execute attribute rules within an edit operation



The Road Ahead



The Road Ahead

- Non-spatial object support (telecom and underground electric)
- Filter barriers
- More control on how features are updated using Update Subnetwork



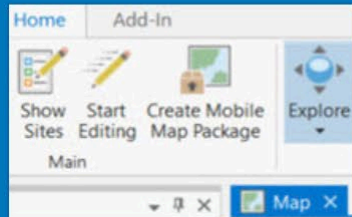
Learning More



Conceptual Doc

<https://pro.arcgis.com/en/pro-app/sdk/>

Learn key concepts



Framework

Build add-ins which extend the Pro UI and leverage asynchronous programming.

[Read the topic](#)



Editing

Create custom tools to construct and edit feature classes with your unique logic.

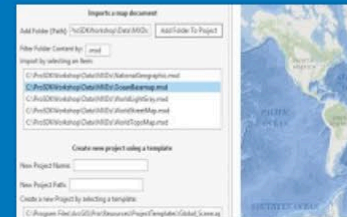
[Read the topic](#)



Map exploration

Create capabilities to navigate and explore the map in 2D and 3D.

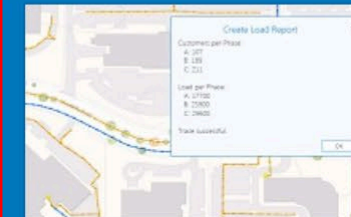
[Read the topic](#)



Content

Manage Pro project content items such as maps, layouts, styles, and more.

[Read the topic](#)



Utility Network

Develop custom network traces, editing tools and diagrams.

[Read the topic](#)

Conceptual Doc

ProConcepts Utility Network

Uma Harano edited this page 27 days ago · 4 revisions

The utility network is a comprehensive framework of functionality in ArcGIS for modeling utility systems such as electric, gas, water, storm water, wastewater, and telecommunications. This topic provides an introduction to the utility network API. It details the classes and methods that query and edit the utility network. The utility network API is commonly used in conjunction with the geodatabase and editing.

Language: C# and Visual Basic
Subject: Utility Network
Contributor: ArcGIS Pro SDK Team <arcgisprosdk@esri.com>
Organization: Esri, <http://www.esri.com>
Date: 1/7/2019
ArcGIS Pro: 2.3
Visual Studio: 2015, 2017

In this topic

- [Introduction](#)
 - [Overview](#)
 - [Resource Management](#)
 - [Namespaces and Extension Methods](#)

► Pages **109**

Home

- [Samples](#)
- [ProSnippets](#)

Developing with ArcGIS Pro

- [Requirements](#)
- [Installing ArcGIS Pro SDK for .NET](#)
- [Getting started](#)
- [ProGuide: ArcGIS Pro Extensions NuGet](#)
- [ProConcepts: Migrating to ArcGIS Pro](#)
- [ProSnippets](#)
- [ArcGIS Pro API](#)
- [Release notes](#)
- [Resources](#)

Samples

ProConcepts Utility Network

Uma Harano edited this page 27 days ago · 4 revisions

The utility network is a comprehensive framework of functionality in ArcGIS for modeling utility systems such as electric, gas, water, storm water, wastewater, and telecommunications. This topic provides an introduction to the utility network API. It details the classes and methods that query and edit the utility network. The utility network API is commonly used in conjunction with the geodatabase and editing.

Language: C# and Visual Basic
Subject: Utility Network
Contributor: ArcGIS Pro SDK Team <arcgisprosdk@esri.com>
Organization: Esri, <http://www.esri.com>
Date: 1/7/2019
ArcGIS Pro: 2.3
Visual Studio: 2015, 2017

In this topic

- [Introduction](#)
 - [Overview](#)
 - [Resource Management](#)
 - [Namespaces and Extension Methods](#)

► Pages **109**

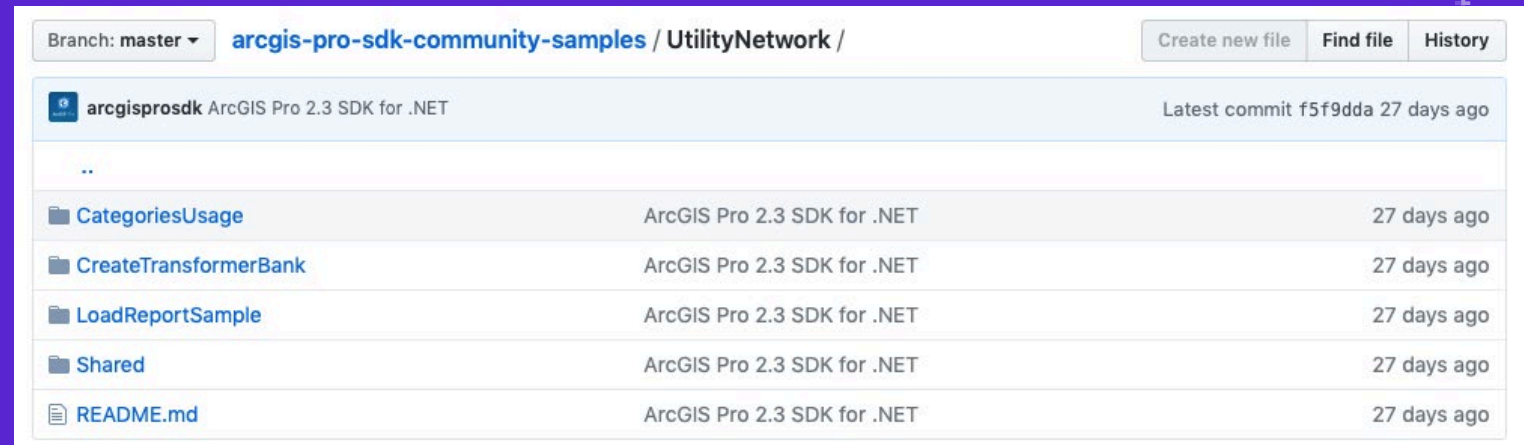
Home

- [Samples](#)

Developing with ArcGIS Pro

- [Requirements](#)
- [Installing ArcGIS Pro SDK for .NET](#)
- [Getting started](#)
- [ProGuide: ArcGIS Pro Extensions NuGet](#)
- [ProConcepts: Migrating to ArcGIS Pro](#)
- [ProSnippets](#)
- [ArcGIS Pro API](#)
- [Release notes](#)
- [Resources](#)

Samples



The screenshot shows a GitHub repository page for 'arcgis-pro-sdk-community-samples / UtilityNetwork'. The repository is on the 'master' branch. The latest commit is 'f5f9dda' from 'arcgisprosdk' 27 days ago. The repository contains several folders and one file:

Item	Description	Commit Date
..		
CategoriesUsage	ArcGIS Pro 2.3 SDK for .NET	27 days ago
CreateTransformerBank	ArcGIS Pro 2.3 SDK for .NET	27 days ago
LoadReportSample	ArcGIS Pro 2.3 SDK for .NET	27 days ago
Shared	ArcGIS Pro 2.3 SDK for .NET	27 days ago
README.md	ArcGIS Pro 2.3 SDK for .NET	27 days ago

- Categories Usage
 - Demonstrates using the schema classes to list all of the asset types that support a given category
- Load Report Sample
 - Does a downstream trace from a starting point and creates a report that sums the load and counts the customers per phase
- Place a Transformer Bank
 - Shows how to create a transformer bank assembly (including the contents and associations)
- Suggestions?
- Write your own?

Snippets

ProConcepts Utility Network

Uma Harano edited this page 27 days ago · 4 revisions

The utility network is a comprehensive framework of functionality in ArcGIS for modeling utility systems such as electric, gas, water, storm water, wastewater, and telecommunications. This topic provides an introduction to the utility network API. It details the classes and methods that query and edit the utility network. The utility network API is commonly used in conjunction with the geodatabase and editing.

Language: C# and Visual Basic
Subject: Utility Network
Contributor: ArcGIS Pro SDK Team <arcgisprosdk@esri.com>
Organization: Esri, <http://www.esri.com>
Date: 1/7/2019
ArcGIS Pro: 2.3
Visual Studio: 2015, 2017

In this topic

- [Introduction](#)
 - [Overview](#)
 - [Resource Management](#)
 - [Namespaces and Extension Methods](#)

► Pages **109**

Home

- [Samples](#)
- [ProSnippets](#)

Developing with ArcGIS Pro

- [Requirements](#)
- [Installing ArcGIS Pro SDK for .NET](#)
- [Getting started](#)
- [ProGuide: ArcGIS Pro Extensions NuGet](#)
- [ProConcepts: Migrating to ArcGIS Pro](#)
- [ProSnippets](#)
- [ArcGIS Pro API](#)
- [Release notes](#)
- [Resources](#)

Snippets

- Small pieces of code that show how to accomplish a specific task

UtilityNetwork Snippets

- [Get a Utility Network from a Table](#)
- [Get a Utility Network from a Layer](#)
- [Find a Tier given a Domain Network Name and Tier Name](#)
- [Update all dirty subnetworks in a tier](#)
- [Creating a DownstreamTracer](#)
- [Create a Trace Argument](#)
- [Create a Condition to compare a Network Attribute against a set of values](#)
- [Create a Function](#)
- [Create a FunctionBarrier](#)
- [Creating an output condition](#)
- [Creating a Propagator](#)
- [Using Function Results](#)
- [Get a list of Inconsistent Network Diagrams](#)
- [Retrieving Diagram Elements](#)
- [Changing the Layout of a Network Diagram](#)

Get a Utility Network from a Layer

```
// This routine obtains a utility network from a FeatureLayer, SubtypeGroupLayer,
public static UtilityNetwork GetUtilityNetworkFromLayer(Layer layer)
{
    if (layer is UtilityNetworkLayer)
    {
        UtilityNetworkLayer utilityNetworkLayer = layer as UtilityNetworkLayer;
        return utilityNetworkLayer.GetUtilityNetwork();
    }

    else if (layer is SubtypeGroupLayer)
    {
        CompositeLayer compositeLayer = layer as CompositeLayer;
        return GetUtilityNetworkFromLayer(compositeLayer.Layers.First());
    }

    else if (layer is FeatureLayer)
    {
        FeatureLayer featureLayer = layer as FeatureLayer;
        using (FeatureClass featureClass = featureLayer.GetFeatureClass())
        {
            if (featureClass.IsControllerDatasetSupported())
            {
                IReadOnlyList<Dataset> controllerDatasets = featureClass.GetControllerDat
                foreach (Dataset controllerDataset in controllerDatasets)
                {
                    if (controllerDataset is UtilityNetwork)
                    {
                        return controllerDataset as UtilityNetwork;
                    }
                }
            }
        }
    }
    return null;
}
```

API Reference

ArcGIS Pro

Home Get Started Help Tool Reference Python SDK

ArcGIS Pro SDK for Microsoft .NET

Extend ArcGIS Pro using the ArcGIS Pro SDK for Microsoft .NET. Develop add-ins and solution configurations to create a custom Pro UI and user experience for your organization. [Download](#) within Visual Studio or at My Esri.

Released Version: 2.3 (January 2019) | [Installation](#) | [What's new in 2.3](#) | [API Reference](#) | [Community Samples](#) | [Documentation](#)

API Reference

Table of Contents

- Introduction
- ArcGIS.Core Assembly
 - Overview
 - Namespaces
 - ArcGIS.Core Namespace
 - ArcGIS.Core.CIM Namespace
 - ArcGIS.Core.Data Namespace
 - Overview
 - Classes
 - Enumerations
 - ArcGIS.Core.Data.Mapping Namespace
 - ArcGIS.Core.Data.PluginDatastore Namespace
 - ArcGIS.Core.Data.Raster Namespace
 - ArcGIS.Core.Data.UtilityNetwork Namespace
 - Overview
 - Classes
 - Enumerations
 - ArcGIS.Core.Data.UtilityNetwork.NetworkDiagrams Namespace
 - ArcGIS.Core.Data.UtilityNetwork.Trace Namespace
 - ArcGIS.Core.Events Namespace
 - ArcGIS.Core.Geometry Namespace

ArcGIS Pro 2.4 API Reference Guide

ArcGIS.Core.Data.UtilityNetwork Namespace

[Inheritance Hierarchy](#) ▶ [Collapse All](#)

This namespace contains types used by the utility network.

Classes

Class	Description
AssetGroup	The AssetGroup class provides information about Asset Groups within the utility network. In the core geodatabase, they are implemented as subtypes.
AssetType	Gets information about the definition of an Asset Type.
Association	Represents a connectivity, containment, or structural attachment association.
AssociationFeature	Represents a connectivity, containment, or structural attachment association, including a ArcGIS.Core.Geometry.Geometry representing a connection between the two rows involved in the association.
ConfigurationPath	The configuration path class details the set of flow paths between Terminals for a given device configuration.
DomainNetwork	The DomainNetwork class is used to represent a domain network inside a utility network. A domain network typically represents an industry domain such as 'Electric Distribution', 'Gas', or 'Water.' DomainNetwork objects can be obtained by calling UtilityNetworkDefinition.GetDomainNetworks .
Element	Represents a row inside a utility network.
NetworkAttribute	The NetworkAttribute class is used to represent a network attribute inside a utility network. Network attributes correspond to weights in the geometric network. NetworkAttribute objects can be obtained by calling

Object Model Diagram

- Finally, [Pro integration](#) describes how the utility network API integrates with other parts of the Pro SDK.

An object model diagram of the utility network API is provided [on the website](#).

UtilityNetwork Class

Utility networks are implemented in the geodatabase as controller datasets. Other controller datasets in ArcGIS include network datasets for transportation networks, and topology for managing coincident features.

The `UtilityNetwork` class provides an abstraction of this controller dataset. Methods on this class provide an entry point to the other areas of the utility network API.

As with other datasets in the geodatabase, a `UtilityNetwork` object can be obtained by calling `Geodatabase.OpenDataset()`. `UtilityNetwork` objects can also be obtained from a table or feature class that belongs to a utility network by using `Table.GetControllerDatasets()`. Note that a particular feature class can belong to multiple controller datasets.

The `Geodatabase.OpenDataset()` routine takes the name of a dataset to open. Remember that when using feature services, the name of the dataset in the feature service workspace does not match the name in the client-server workspace. You typically need to get the

Tasks

- [ProSnippets: Tasks](#)
- [ProConcepts: Tasks](#)

Utility Network

- [ProSnippets: Utility Network](#)
- [ProConcepts: Utility Network](#)
- [Object Model Diagram](#)

Workflow Manager

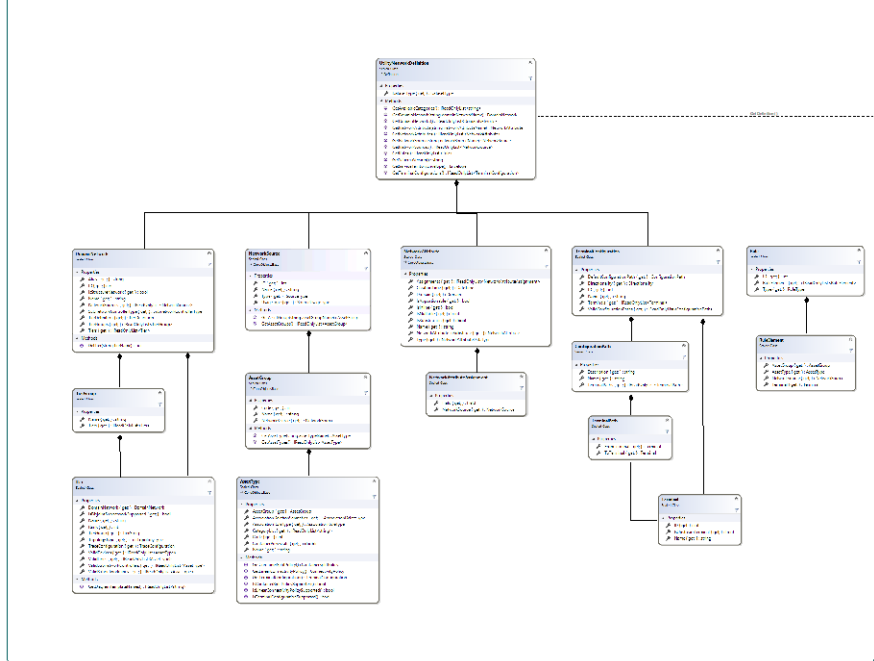
- [ProSnippets: Workflow Manager](#)
- [ProConcepts: Workflow Manager](#)

Utility Network Object Model

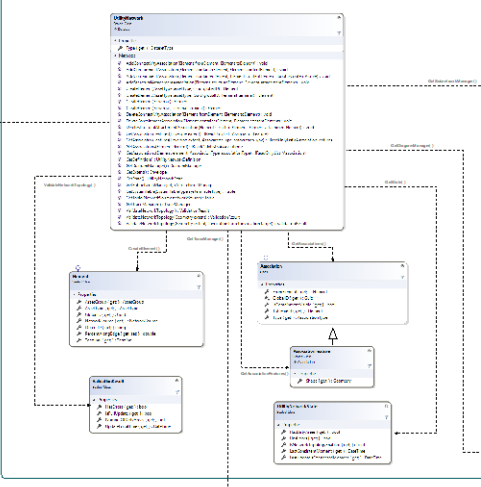
ArcGIS Pro 2.3

Copyright © 2019 Esri, Inc.

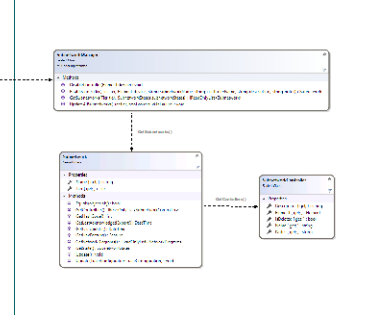
Schema



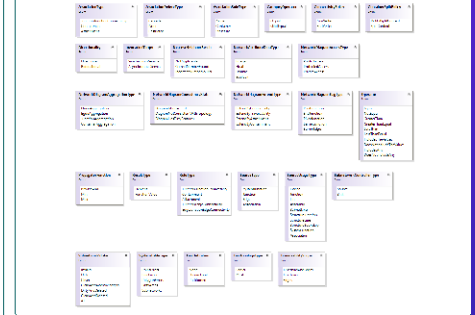
Utility Network Core



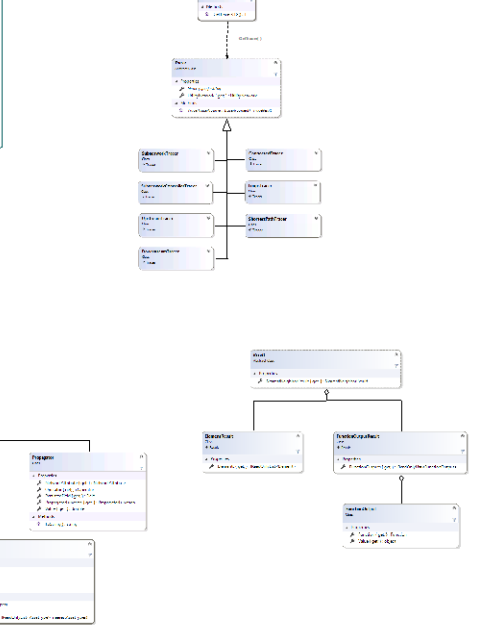
Subnetwork



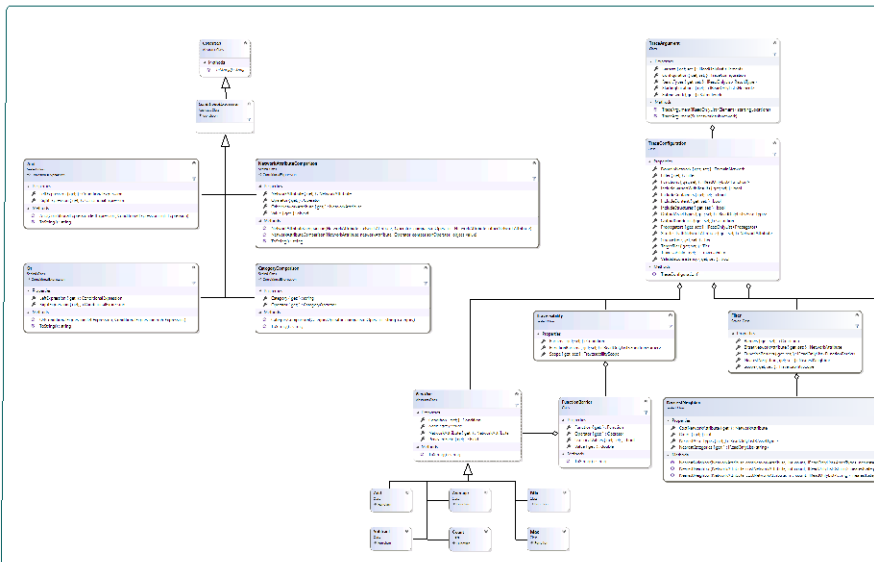
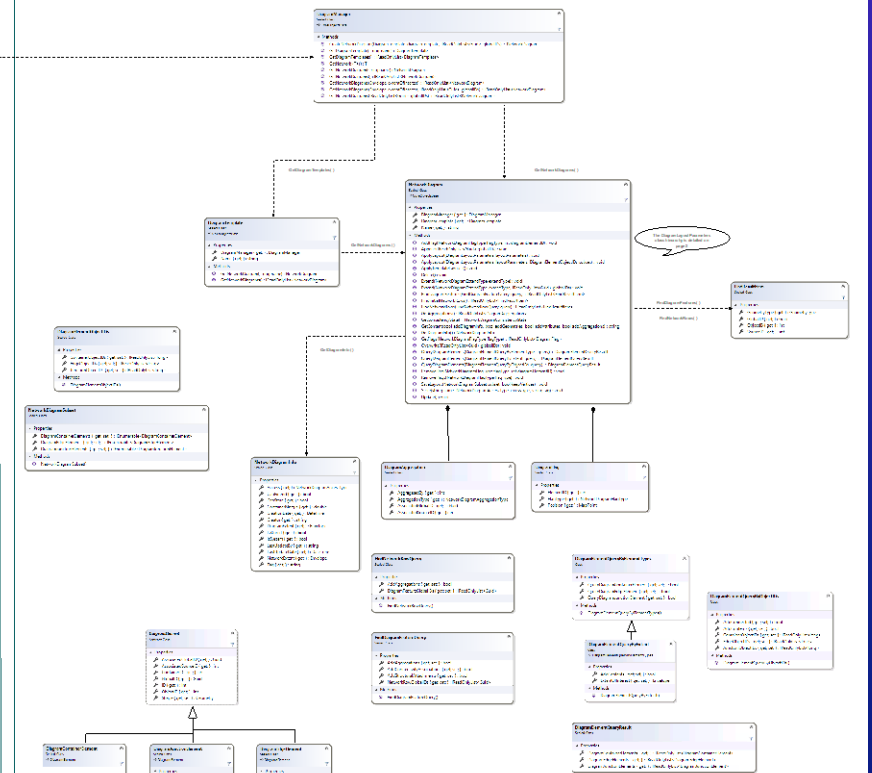
Enumerations



Tracing



Network Diagrams



An Overview of the Utility Network Management API

- **Session materials are available here:**

- <https://github.com/esri/arcgis-pro-sdk/wiki/tech-sessions#2020-palm-springs>

