



ArcGIS Runtime: Utilities

Rich Ruh

Jennifer Nery

Tony Wakim

2021 ESRI
DEVELOPER SUMMIT

The Utility Network

- The utility network is a cross-platform Esri technology for modeling utility and telecom networks
- Create client-server database using ArcGIS Pro
- Publish utility network and feature services to ArcGIS Enterprise
- Query and edit data on multiple platforms
 - ArcGIS Pro
 - Web
 - ArcGIS Runtime

Today's Topics

- Core concepts
 - Two foundational classes
 - Schema information
 - Associations
 - Tracing
- Online capabilities
 - Utility networks with ArcGIS Enterprise
- Offline capabilities
 - Utility networks with mobile geodatabases
- The road ahead
 - New capabilities that will be available later this year and beyond
- This presentation assumes a basic familiarity with the utility network information model
 - [Network Management with ArcGIS – Introduction to the Utility Network \(2020\)](#)



Two Foundational Classes

Core Concepts

The Utility Network Class

- Serves as a central hub of the utility network API
- How do you get a `UtilityNetwork` object?
 - Create from a feature service URL and a map
 - Obtain it from a map directly
 - From a mobile geodatabase (not yet)

The Utility Element Class

- The **UtilityElement** 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 retrieve associations
 - Elements specify starting points and barriers for use with tracing
 - Elements are returned as results from traces
- Created using **CreateElement()** factory methods on the **UtilityNetwork** class

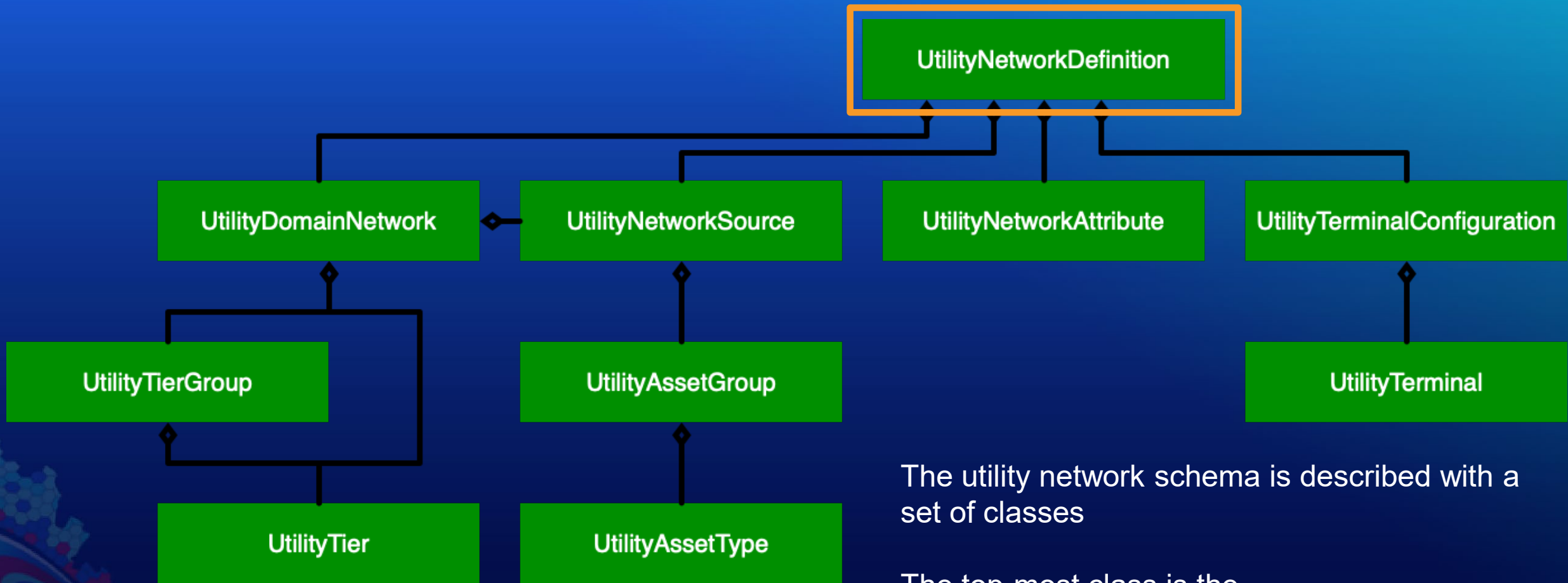
<code>createElement(ArcGISFeature arcGISFeature)</code>	Creates a UtilityElement from an ArcGISFeature .
<code>createElement(ArcGISFeature arcGISFeature, UtilityTerminal terminal)</code>	Creates a UtilityElement from an ArcGISFeature and an optional UtilityTerminal .
<code>createElement(UtilityAssetType assetType, UUID globalId)</code>	Creates a UtilityElement from a UtilityAssetType and a global ID.
<code>createElement(UtilityAssetType assetType, UUID globalId, UtilityTerminal terminal)</code>	Creates a UtilityElement from a UtilityAssetType , a global ID and an optional UtilityTerminal .



Schema Information

Core Concepts

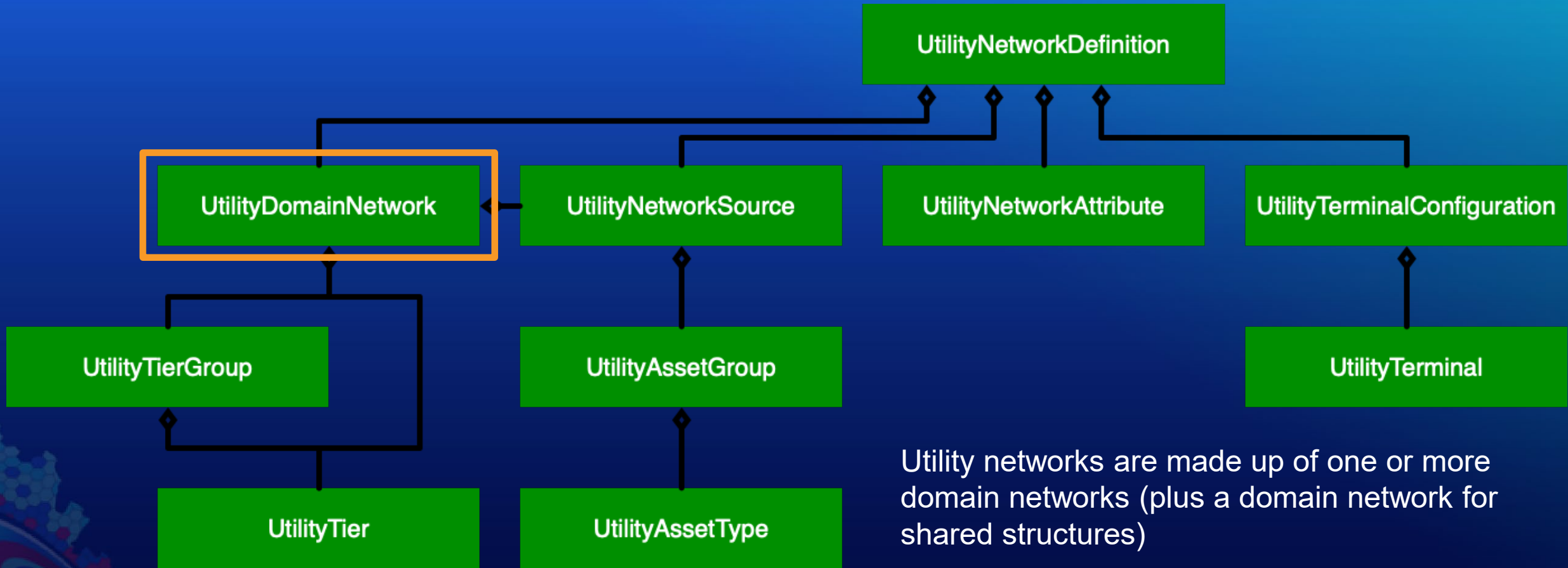
Utility Network Schema Information



The utility network schema is described with a set of classes

The top-most class is the `UtilityNetworkDefinition`, which is obtained using `UtilityNetwork.Definition`

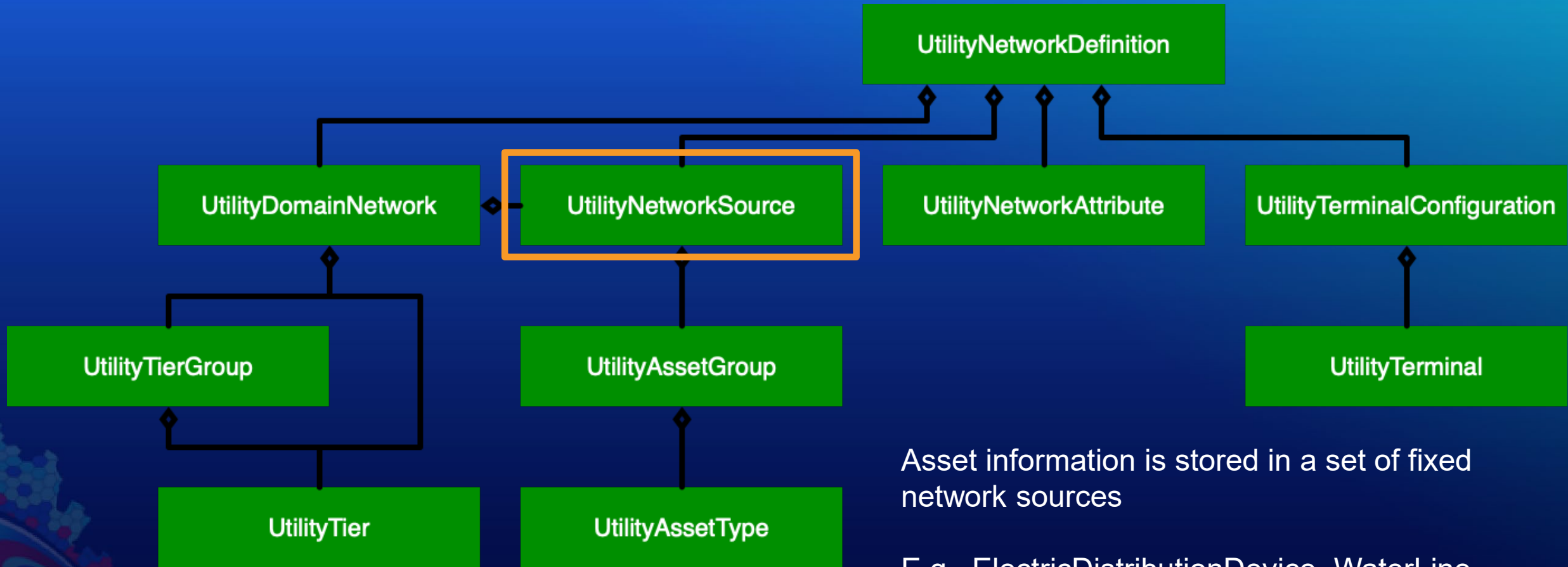
Utility Network Schema Information



Utility networks are made up of one or more domain networks (plus a domain network for shared structures)

E.g., electric distribution, water, telecom

Utility Network Schema Information

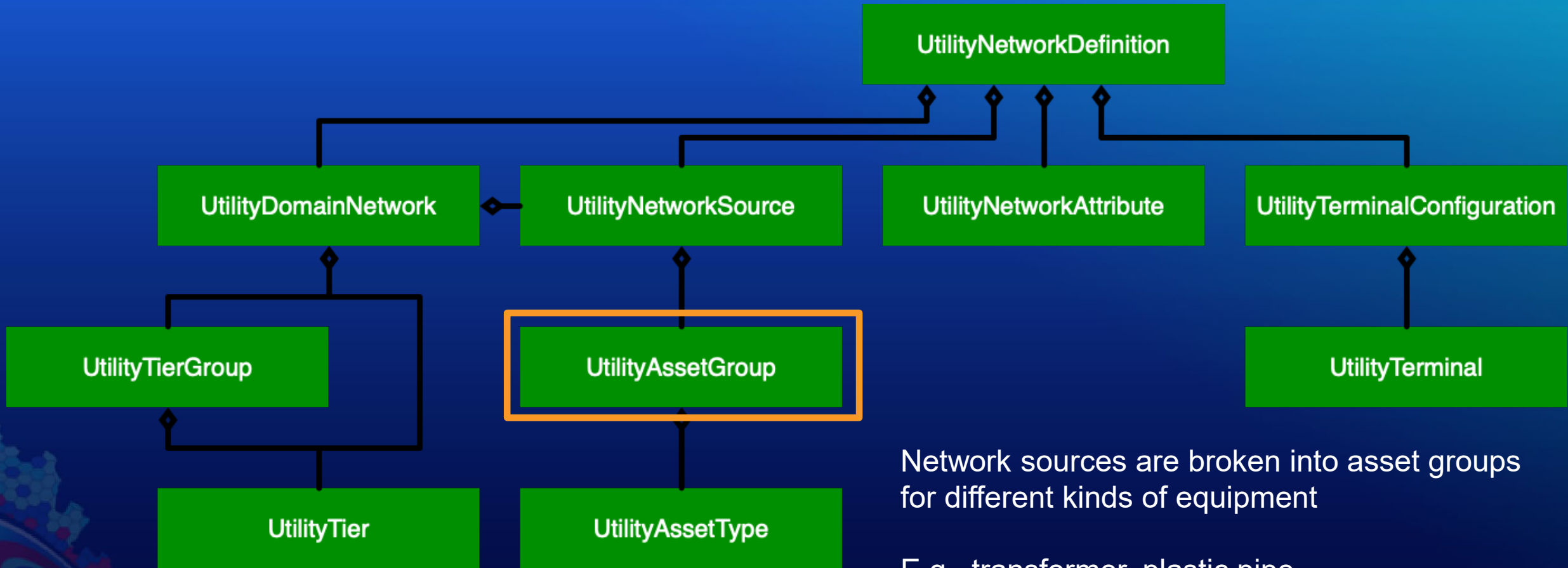


Asset information is stored in a set of fixed network sources

E.g., ElectricDistributionDevice, WaterLine

Network sources = feature tables

Utility Network Schema Information

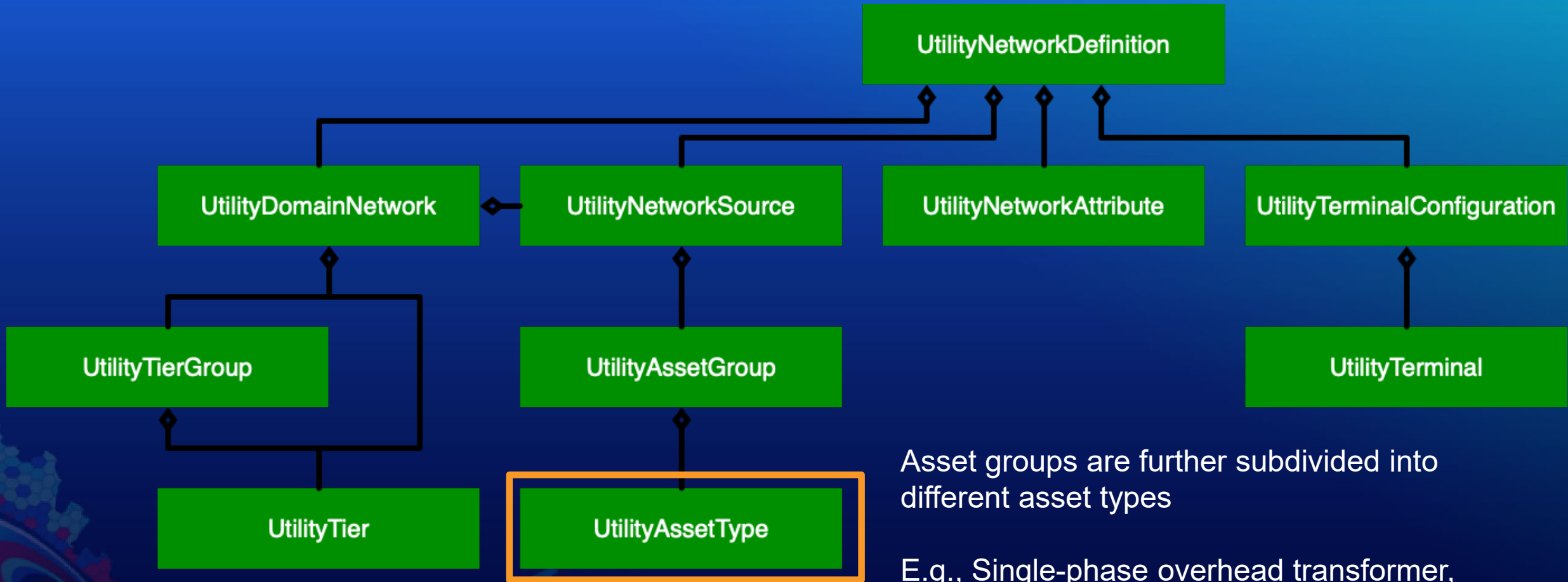


Network sources are broken into asset groups for different kinds of equipment

E.g., transformer, plastic pipe

Implemented in the geodatabase as subtypes

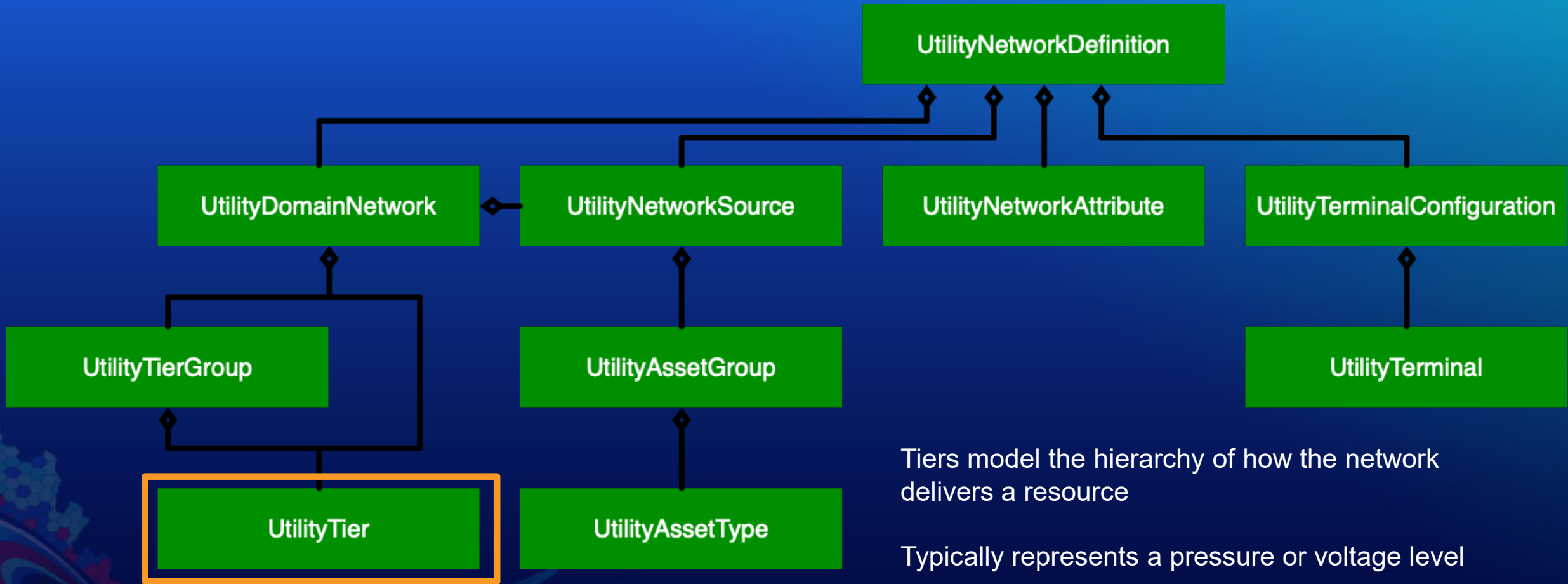
Utility Network Schema Information



Asset groups are further subdivided into different asset types

E.g., Single-phase overhead transformer, 4" plastic pipe

Utility Network Schema Information

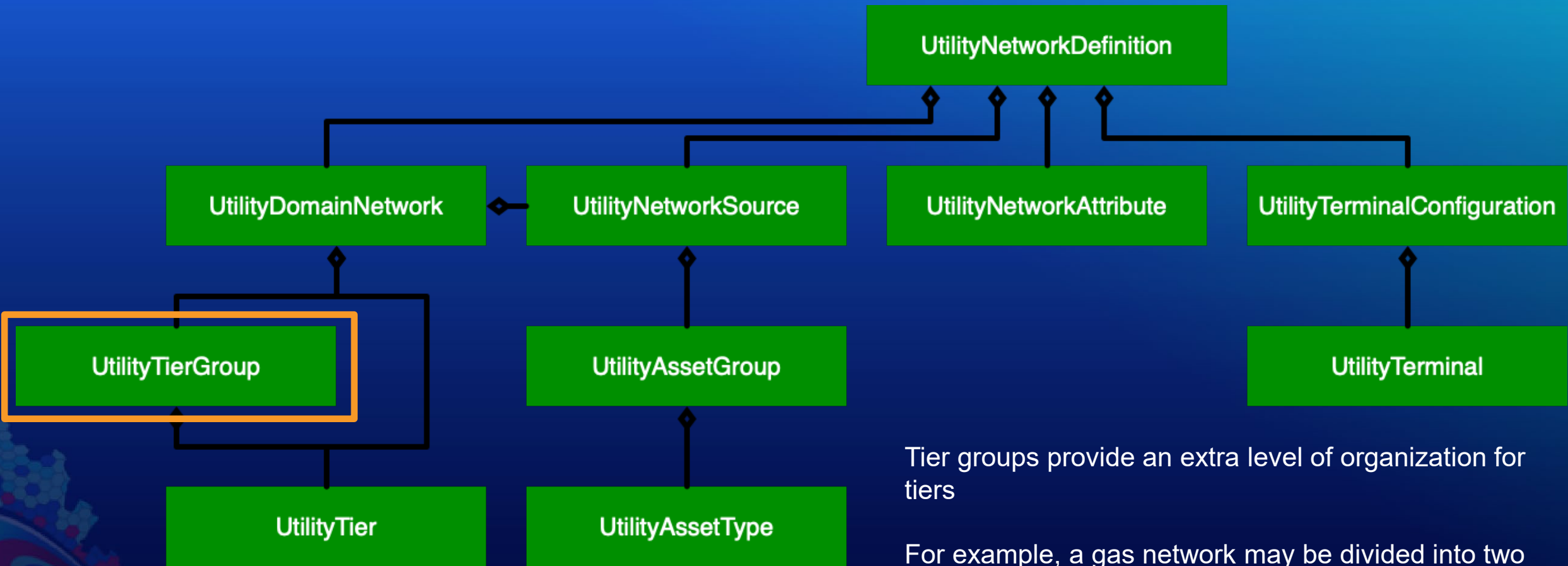


Tiers model the hierarchy of how the network delivers a resource

Typically represents a pressure or voltage level

Can also represent parts of the network that can be isolated from one another

Utility Network Schema Information

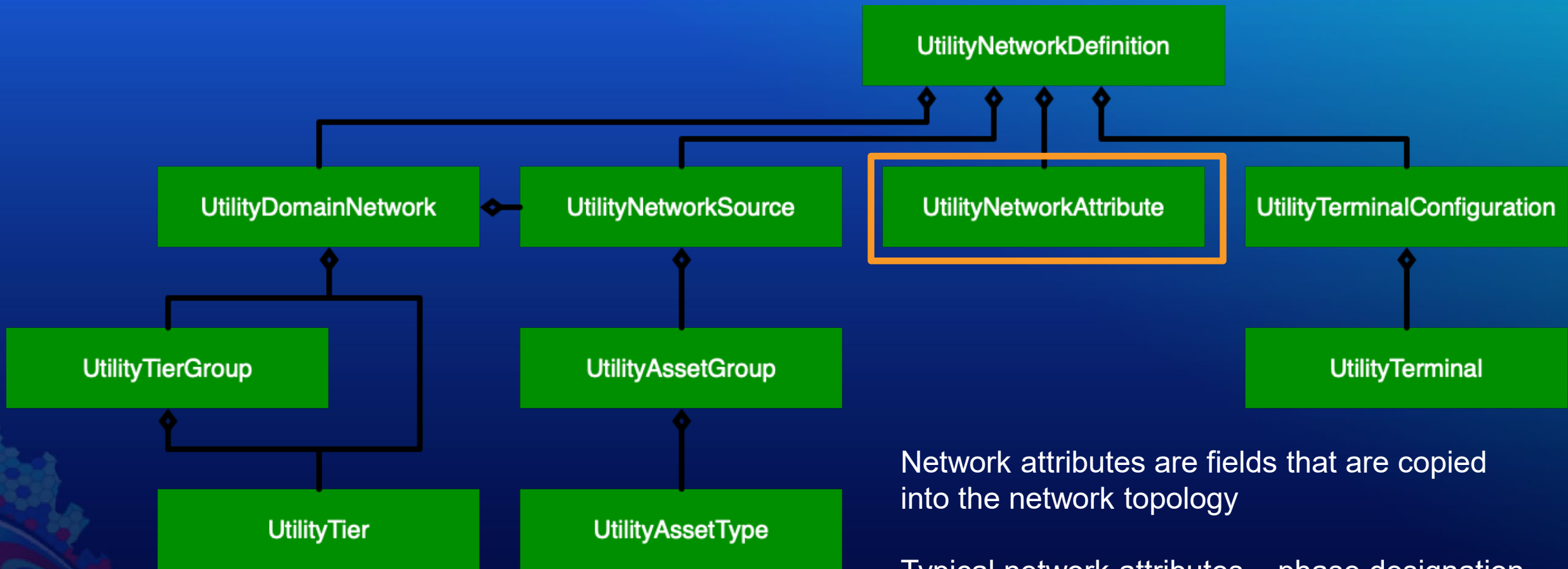


Tier groups provide an extra level of organization for tiers

For example, a gas network may be divided into two tier groups – transmission and distribution

Each of these tier groups would contain a set of tiers specific to that group

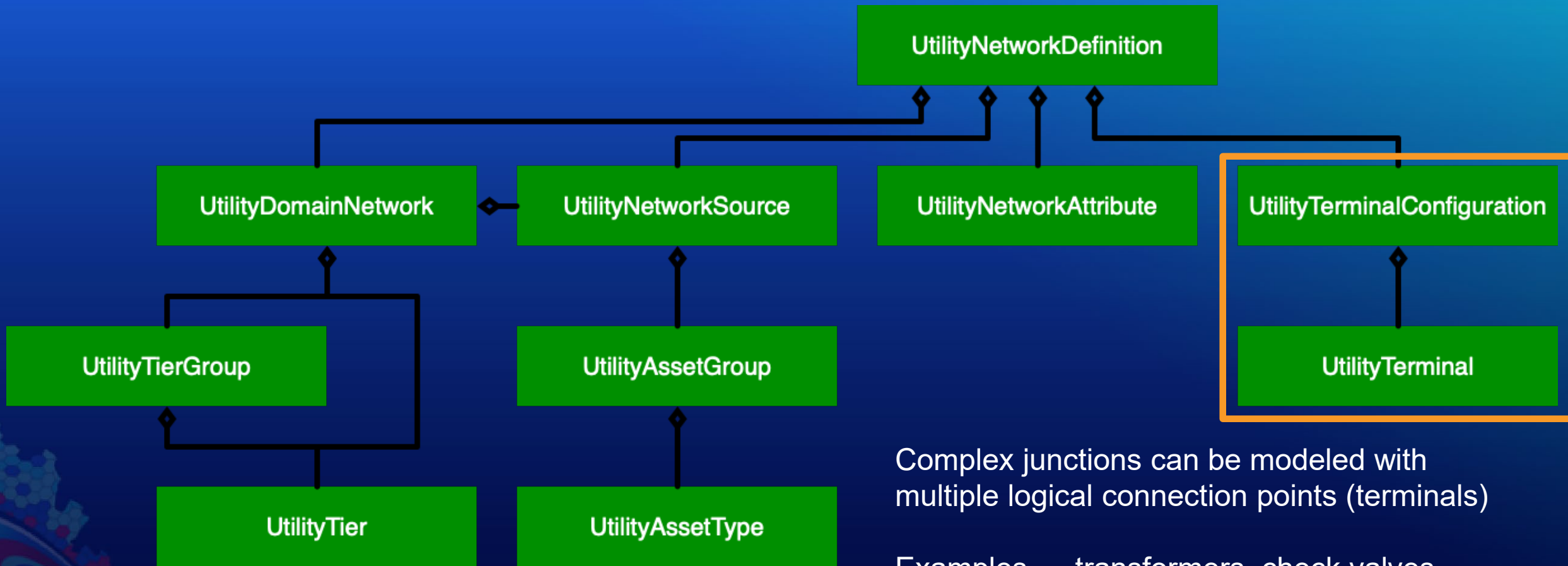
Utility Network Schema Information



Network attributes are fields that are copied into the network topology

Typical network attributes – phase designation, valve position, pipe material

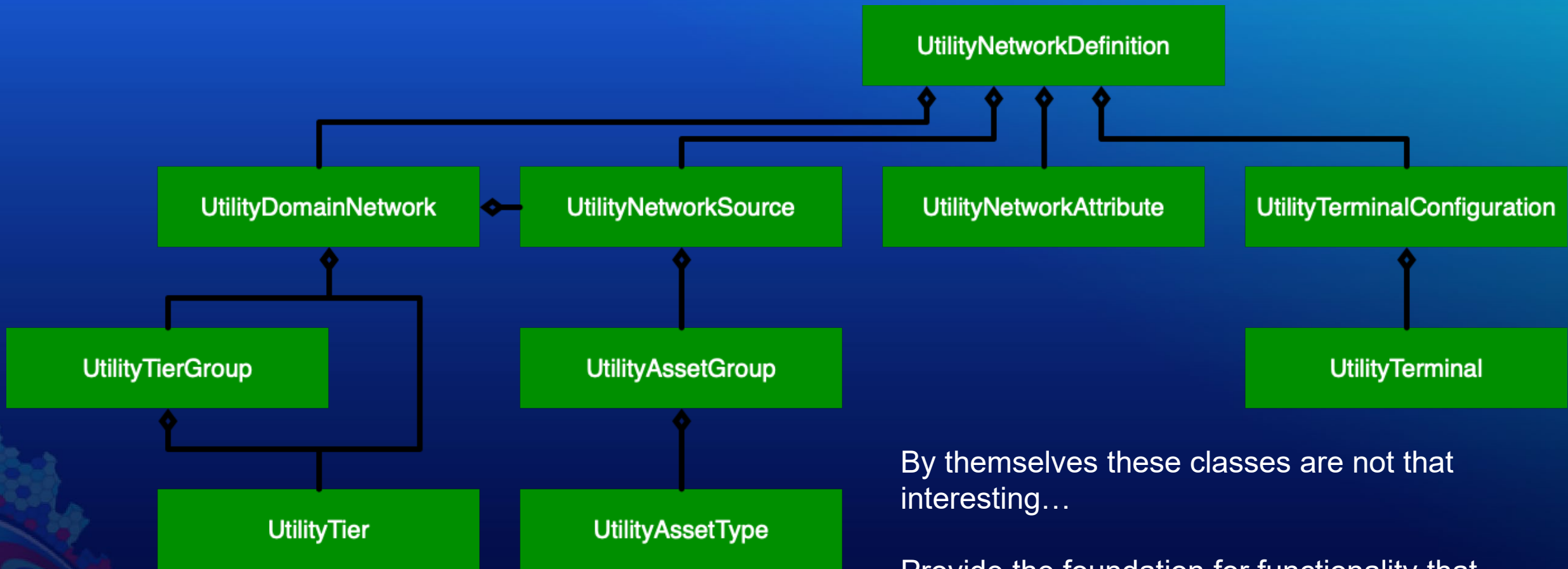
Utility Network Schema Information



Complex junctions can be modeled with multiple logical connection points (terminals)

Examples — transformers, check valves

Utility Network Schema Information



By themselves these classes are not that interesting...

Provide the foundation for functionality that uses the data model



Associations

Concepts

Associations

- Associations are connections between features (or parts of features) that are stored in the network topology
- Connectivity associations
 - Connect two features without requiring coincident geometry
 - Service points connected to low-voltage terminal on a transformer
- Structural attachment associations
 - Models a physical attachment between a piece of equipment (transformer) and a structure (pole)
- Containment associations
 - Allows content (fuses, transformers, arrestors) to be stored inside a container (transformer bank)

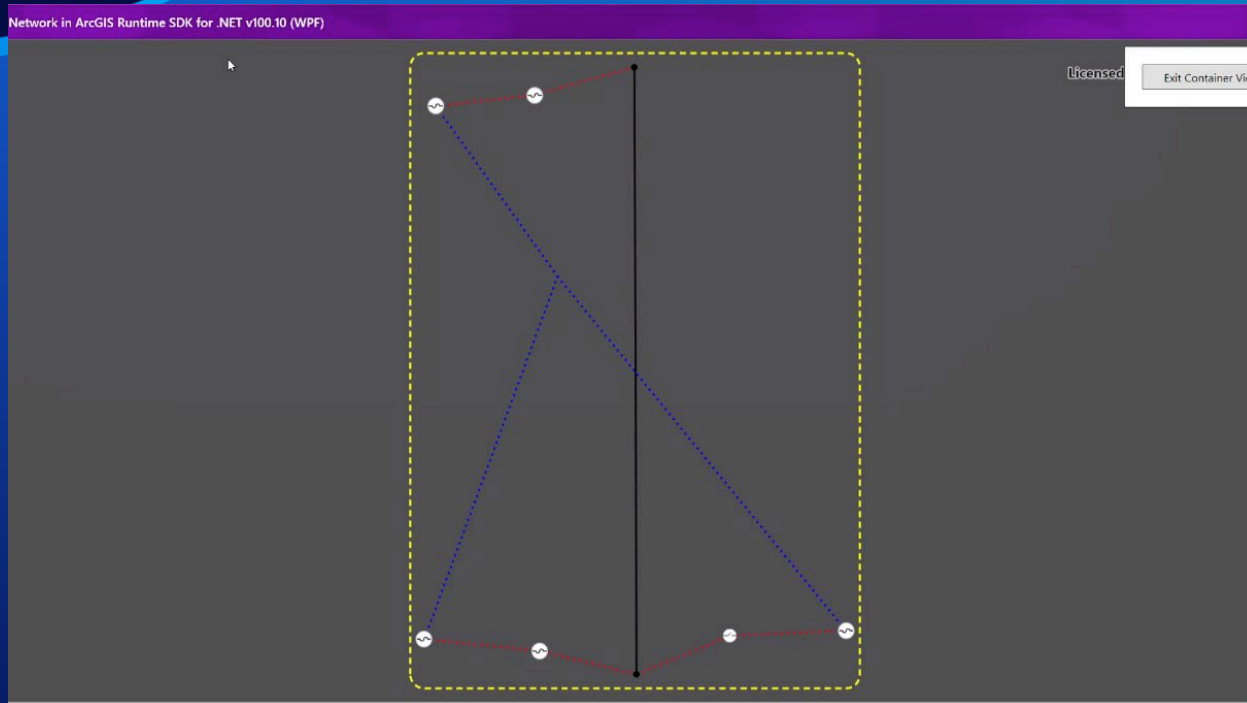
Association Queries

- Get associations from a particular element
 - What features are inside this container?

`GetAssociationsAsync(UtilityElement)` Asynchronously retrieves a list of all `UtilityAssociations` in the geodatabase for a given `UtilityElement`.

- Get associations from an envelope
 - What connectivity and structural attachment associations are within our map extent?

`GetAssociationsAsync(Envelope)` Asynchronously retrieves a list of all `UtilityAssociation` objects (with their geometry) present in the geodatabase for a given *extent*.



Associations

Jennifer Nery

[JSON](#) | [SOAP](#)

UtilityNetwork/NapervilleElectric (FeatureServer)

View In: [ArcGIS Online Map Viewer](#)

View Footprint In: [ArcGIS Online Map Viewer](#)

Service Description:

This service has been designed to support the advance capabilities of the ArcGIS Utility Network Management extension. It is not designed to be used without additional layer design provided through ArcGIS Pro or a webmap.

[All Layers and Tables](#)

Has Versioned Data: true

MaxRecordCount: 2000

Supported Query Formats: JSON

Supports Query Data Elements: true

[Dynamic Legend](#)

[Dynamic All Layers](#)

Layers:

- [Electric Distribution Device](#) (100)
- [Electric Distribution Assembly](#) (105)
- [Electric Distribution Junction](#) (110)
- [Electric Distribution Line](#) (115)
- [Electric Distribution SubnetLine](#) (120)
- [Structure Junction](#) (900)
- [Structure Line](#) (905)
- [Structure Boundary](#) (910)
- [Service Territory](#) (920)
- [Electric Utility Network](#) (921)
 - [Point Errors](#) (0)
 - [Line Errors](#) (1)
 - [Polygon Errors](#) (2)
 - [Dirty Areas](#) (3)

Tables:

- [Joint Use](#) (1500)

Description:

Copyright Text: Esri., Inc.

Contents

Search



Drawing Order

- Naperville Electric Containers
 - World Dark Gray Reference
 - Electric Distribution Assembly
 - Unknown
 - Arrester Bank
 - Capacitor Bank
 - Circuit Breaker Bank
 - Fuse Bank
 - Switch Bank
 - Transformer Bank
 - Voltage Regulator Bank
 - Service Location
 - Structure Junction
 - World Dark Gray Canvas Base



Enter container view

```
_previousViewpoint = MyMapView.GetCurrentViewpoint(ViewpointType.BoundingGeometry);  
  
foreach (Layer layer in MyMapView.Map.OperationalLayers)  
    layer.IsVisible = false;  
  
ContainerViewControl.Visibility = Visibility.Visible;  
  
GraphicsOverlay overlay = MyMapView.GraphicsOverlays[0];
```


Identify container feature

```
private async void OnGeoViewTapped(object sender, GeoViewInputEventArgs e)
{
    IReadOnlyList<IdentifyLayerResult> identifyLayerResults =
        await MyMapView.IdentifyLayersAsync(e.Position, 5, false);

    foreach (IdentifyLayerResult layerResult in identifyLayerResults)
    {
        if (_containerFeature == null && layerResult.LayerContent is SubtypeFeatureLayer)
        {
            foreach (IdentifyLayerResult subLayerResult in layerResult.SublayerResults)
            {
                foreach (GeoElement geoElement in subLayerResult.GeoElements)
                {
                    if (_containerFeature == null && geoElement is ArcGISFeature feature)
                    {
                        _containerFeature = feature;
                        break;
                    }
                }
            }
        }
    }
}
```

Inspect containment associations

```
UtilityElement containerElement = _utilityNetwork.CreateElement(_containerFeature);

IEnumerable<UtilityAssociation> containmentAssociations =
    await _utilityNetwork.GetAssociationsAsync(containerElement, UtilityAssociationType.Containment);

List<UtilityElement> contentElements = new List<UtilityElement>();
foreach (UtilityAssociation association in containmentAssociations)
{
    UtilityElement otherElement = association.FromElement.ObjectId == containerElement.ObjectId
        ? association.ToElement : association.FromElement;
    contentElements.Add(otherElement);
}
```

Display content features

```
IEnumerable<ArcGISFeature> contentFeatures =  
    await _utilityNetwork.GetFeaturesForElementsAsync(contentElements);  
  
foreach (ArcGISFeature content in contentFeatures)  
{  
    Symbol symbol = ((ArcGISFeatureTable)content.FeatureTable).  
        LayerInfo.DrawingInfo.Renderer.GetSymbol(content);  
    overlay.Graphics.Add(new Graphic(content.Geometry, symbol));  
}
```

Visualize associations from an area of interest

```
containmentAssociations = await _utilityNetwork.GetAssociationsAsync(overlay.Extent);  
foreach (UtilityAssociation association in containmentAssociations)  
{  
    Symbol symbol = association.AssociationType == UtilityAssociationType.Attachment ?  
        AttachmentSymbol : ConnectivitySymbol;  
    overlay.Graphics.Add(new Graphic(association.Geometry, symbol));  
}
```

Exit container view

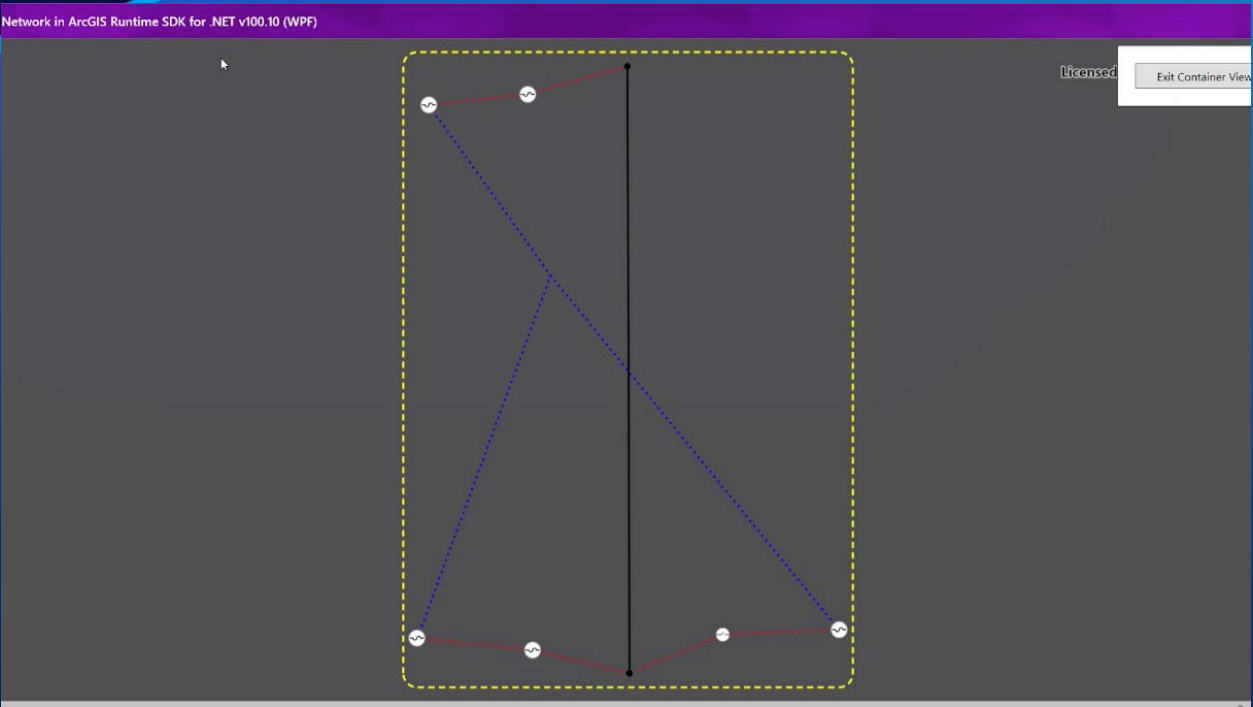
```
ContainerViewControl.Visibility = Visibility.Collapsed;

_containerFeature = null;

MyMapView.GraphicsOverlays[0].Graphics.Clear();

foreach (Layer layer in MyMapView.Map.OperationalLayers)
    layer.IsVisible = true;

if (_previousViewpoint is Viewpoint viewpoint)
    MyMapView.SetViewpointAsync(viewpoint);
```



Associations

Jennifer Nery



Tracing

Core Concepts

Tracing

- Classes to support network analysis
- 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

Runtime Currently Provides a Subset of Utility Network Tracing Functionality

- Trace Types

- Connected
- Subnetwork (based on starting points)
- Upstream
- Downstream
- Isolation
- Subnetwork (based on subnetwork name)
- Subnetwork controller
- Loops
- Shortest path

- Trace Configuration

- Starting points and barriers
- Traversability (network attribute and category comparisons)
- Traversability (function barriers)
- Functions
- Filters
- Output Filters
- Propagators
- Filter Barriers
- Aggregated Geometry

Runtime Currently Provides a Subset of Utility Network Tracing Functionality

- Trace Types

- Connected
- Subnetwork (based on starting points)
- Upstream
- Downstream
- Isolation

Subnetwork (based on subnetwork name)

Subnetwork controller

- Loops
- Shortest path

- Trace Configuration

- Starting points and barriers
- Traversability (network attribute and category comparisons)
- Traversability (function barriers)
- Functions
- Filters
- Output Filters
- Propagators
- Filter Barriers
- Aggregated Geometry

Trace Types

- Connected Trace
- Subnetwork-based Traces
 - Subnetwork
 - Upstream
 - Downstream
- Isolation Trace
- Shortest Path
- Loops

The most basic trace- returns connected features

Trace Types

- Connected Trace
- Subnetwork-based Traces
 - Subnetwork
 - Upstream
 - Downstream
- Isolation Trace
- Shortest Path
- Loops

Subnetworks are collections of connected utility network features

- Represent circuits in electric
- Represent zones in gas, water, wastewater

Typically defined as follows:

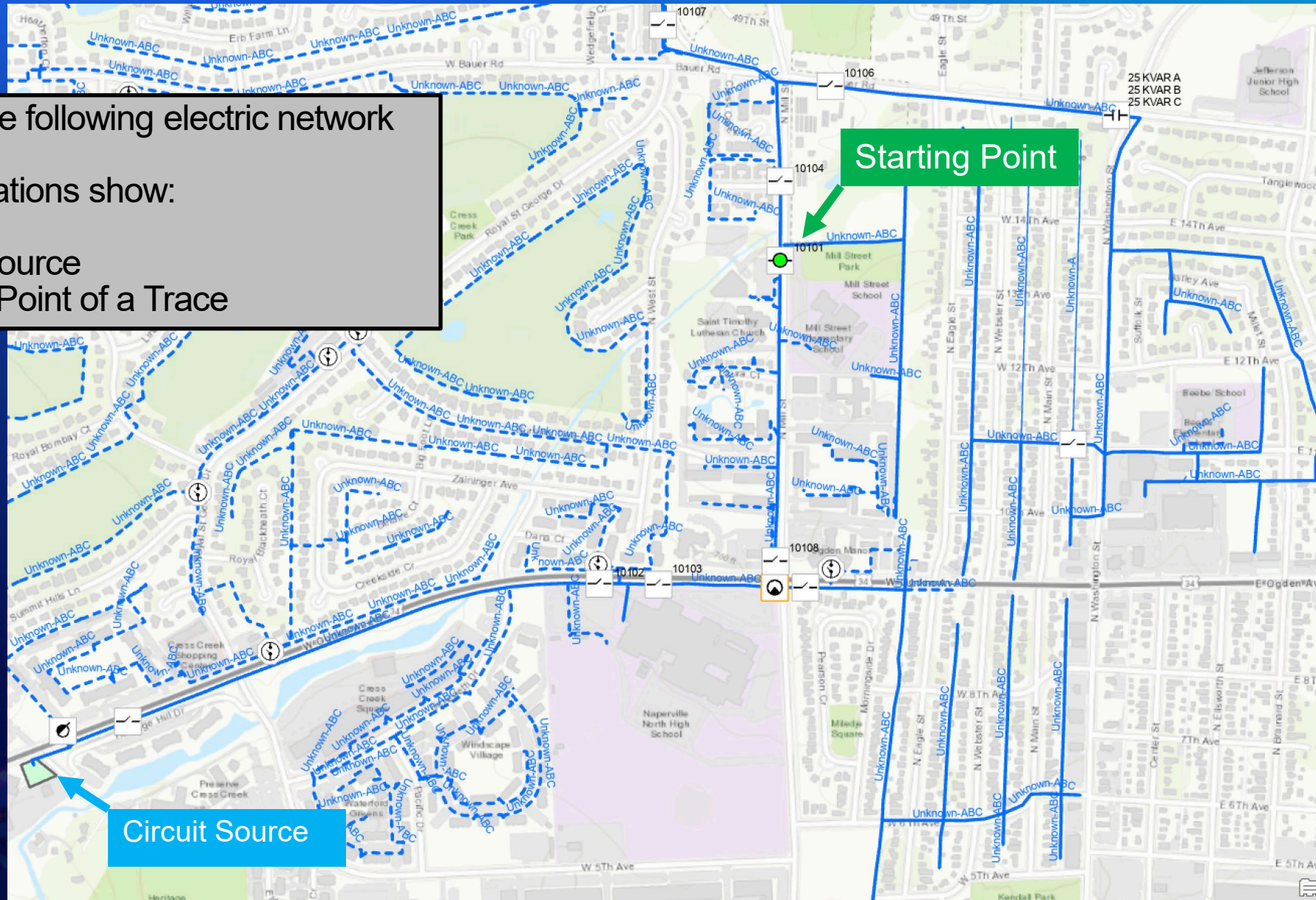
- Start at a source (e.g., circuit breaker) or end at a sink (e.g., sewage treatment plant)
- Based on a trace configuration which defines the extent of the circuit (e.g., gas zones will stop at a closed valve)

Subnetwork-based Tracing

Consider the following electric network

Marked locations show:

- Circuit Source
- Starting Point of a Trace

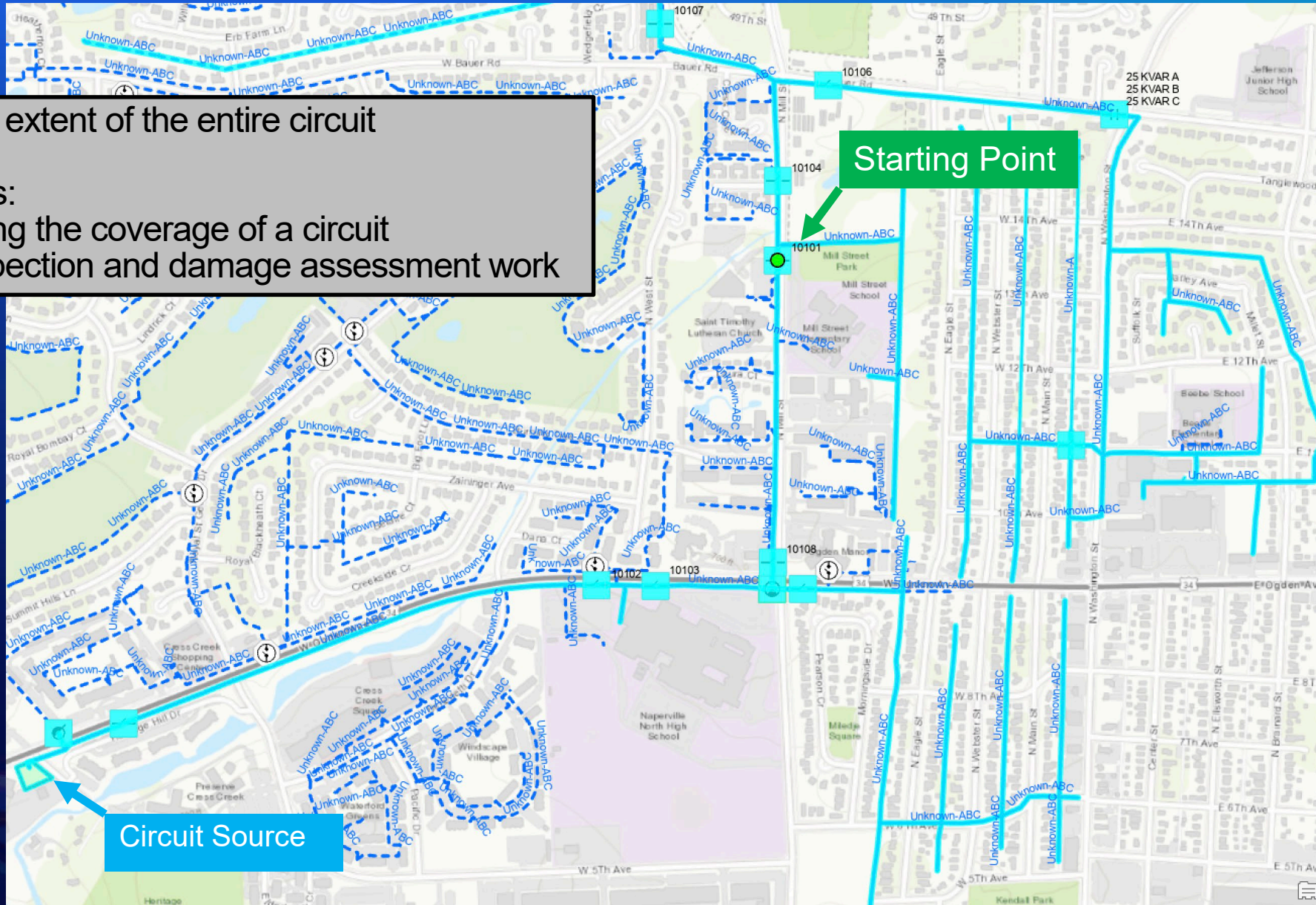


Subnetwork Trace

Returns the extent of the entire circuit

Typical uses:

- Visualizing the coverage of a circuit
- Plan inspection and damage assessment work

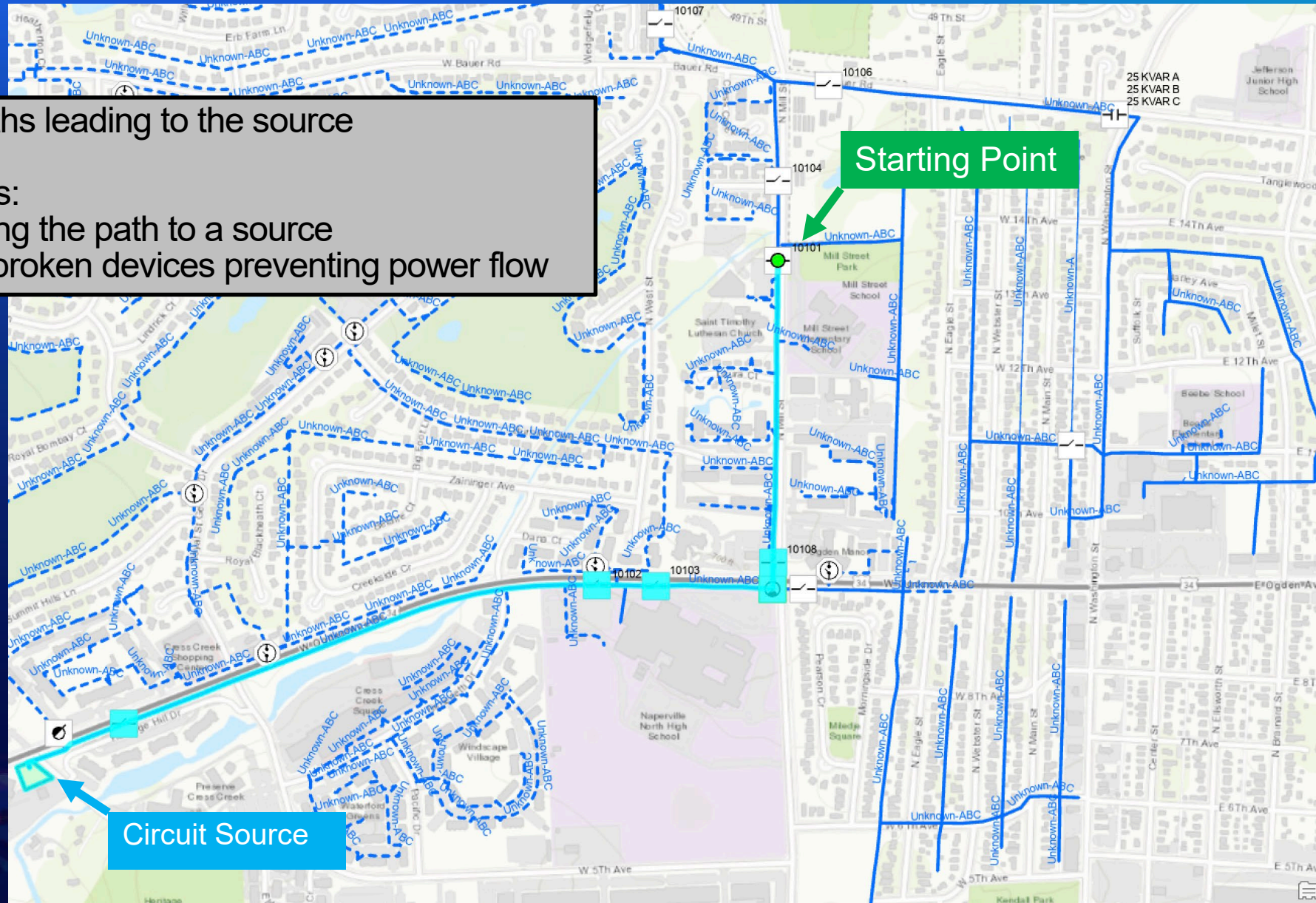


Upstream Trace

Returns paths leading to the source

Typical uses:

- Visualizing the path to a source
- Finding broken devices preventing power flow

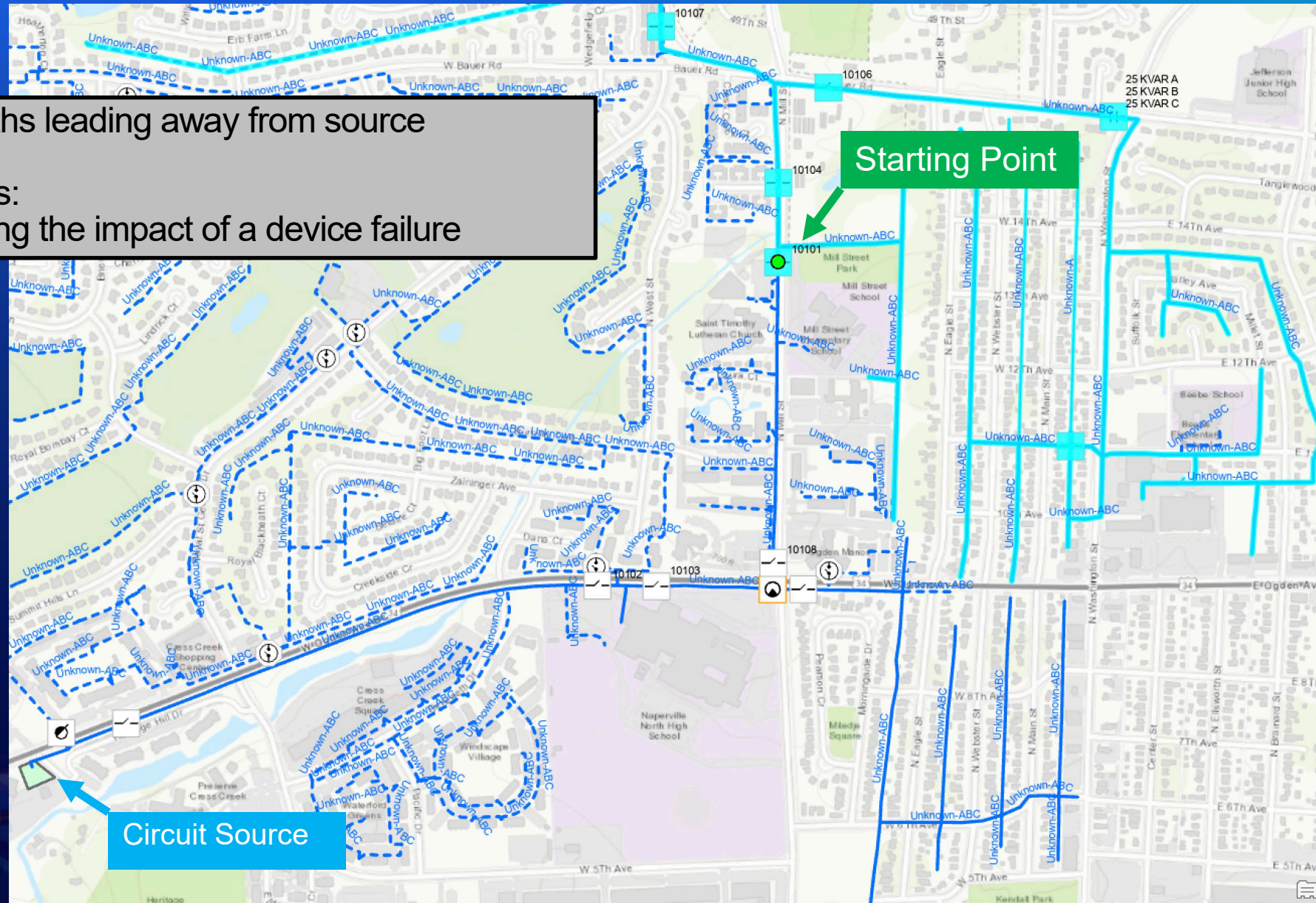


Downstream Trace

Returns paths leading away from source

Typical uses:

- Visualizing the impact of a device failure



Trace Types

- Connected Trace
- Subnetwork-based Traces
 - Subnetwork
 - Upstream
 - Downstream
- Isolation Trace
- Shortest Path
- Loops

Returns devices that must be operated to shut off resource flow to a particular starting point

Can also return the set of features that are isolated if those devices are operated

Trace Types

- Connected Trace
- Subnetwork-based Traces
 - Subnetwork
 - Upstream
 - Downstream
- Isolation Trace
- Shortest Path
- Loops

Returns the shortest path between the starting points

"Shortest" can be defined as geographic distance... or as cost

Trace Types

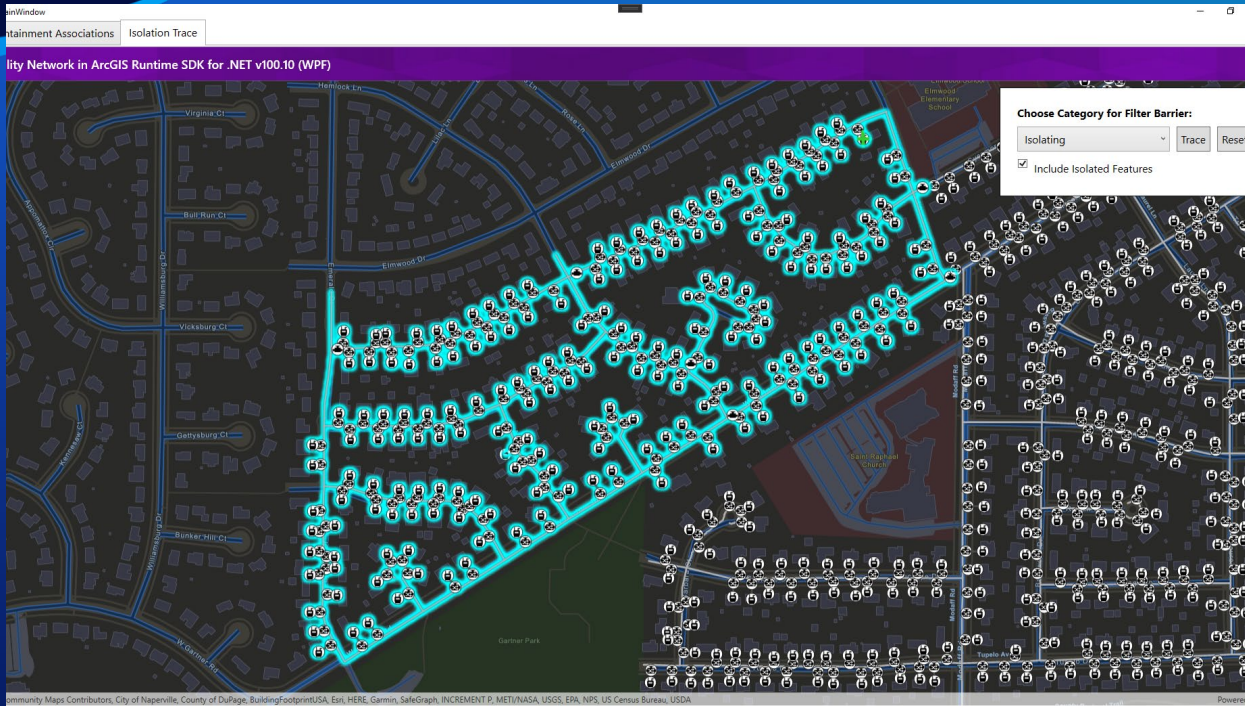
- Connected Trace
- Subnetwork-based Traces
 - Subnetwork
 - Upstream
 - Downstream
- Isolation Trace
- Shortest Path
- Loops

Identifies features that participate in loops

Tracing Configuration

- Starting points, barriers, and filter barriers
- Traversability
- Functions
- Filters
- Output filters
- Propagators

- Too much for one DevSummit talk!
- [Check out Pro SDK talk from 2020](#)



Isolation Trace

Jennifer Nery

Isolation trace parameters

```
_parameters = new UtilityTraceParameters(UtilityTraceType.Isolation,  
                                          new UtilityElement[] { startingLocation });  
  
UtilityDomainNetwork domainNetwork = _utilityNetwork.Definition.GetDomainNetwork(DomainNetworkName);  
UtilityTier tier = domainNetwork.GetTier(TierName);  
_parameters.TraceConfiguration = tier.TraceConfiguration;  
  
_parameters.TraceConfiguration.IncludeIsolatedFeatures = IncludeIsolatedFeatures.IsChecked == true;  
  
if (Categories.SelectedItem is UtilityCategory category)  
{  
    _parameters.TraceConfiguration.Filter = new UtilityTraceFilter()  
    {  
        Barriers = new UtilityCategoryComparison(category,  
                                                UtilityCategoryComparisonOperator.Exists)  
    };  
}
```

Selecting element trace results

```
IEnumerable<UtilityTraceResult> traceResults = await _utilityNetwork.TraceAsync(_parameters);
foreach (UtilityTraceResult traceResult in traceResults)
{
    if(traceResult.Warnings.Count > 0)
    {
        MessageBox.Show(string.Join(Environment.NewLine, traceResult.Warnings));
    }
    if(traceResult is UtilityElementTraceResult elementTraceResult)
    {
        foreach (FeatureLayer layer in MyMapView.Map.OperationalLayers.OfType<FeatureLayer>())
        {
            IEnumerable<UtilityElement> elements = from element in elementTraceResult.Elements
                                                    where element.NetworkSource.FeatureTable
                                                         == layer.FeatureTable
                                                    select element;
            IEnumerable <Feature> features = await _utilityNetwork.GetFeaturesForElementsAsync(elements);
            layer.SelectFeatures(features);
        }
    }
}
```

Viewing geometry trace results

```
_parameters.ResultTypes.Add(UtilityTraceResultType.Geometry);
IEnumerable<UtilityTraceResult> traceResults = await _utilityNetwork.TraceAsync(_parameters);
foreach (UtilityTraceResult traceResult in traceResults)
{
    if(traceResult.Warnings.Count > 0)
    {
        MessageBox.Show(string.Join(Environment.NewLine, traceResult.Warnings));
    }
    if(traceResult is UtilityGeometryTraceResult geometryTraceResult)
    {
        if (geometryTraceResult.Polygon is Polygon polygon)
            graphicsOverlay.Graphics.Add(new Graphic(polygon, polygonSymbol));
        if (geometryTraceResult.Polyline is Polyline polyline)
            graphicsOverlay.Graphics.Add(new Graphic(polyline, polylineSymbol));
        if (geometryTraceResult.Multipoint is Multipoint multipoint)
            graphicsOverlay.Graphics.Add(new Graphic(multipoint, pointSymbol));
    }
}
```


Viewing function trace results

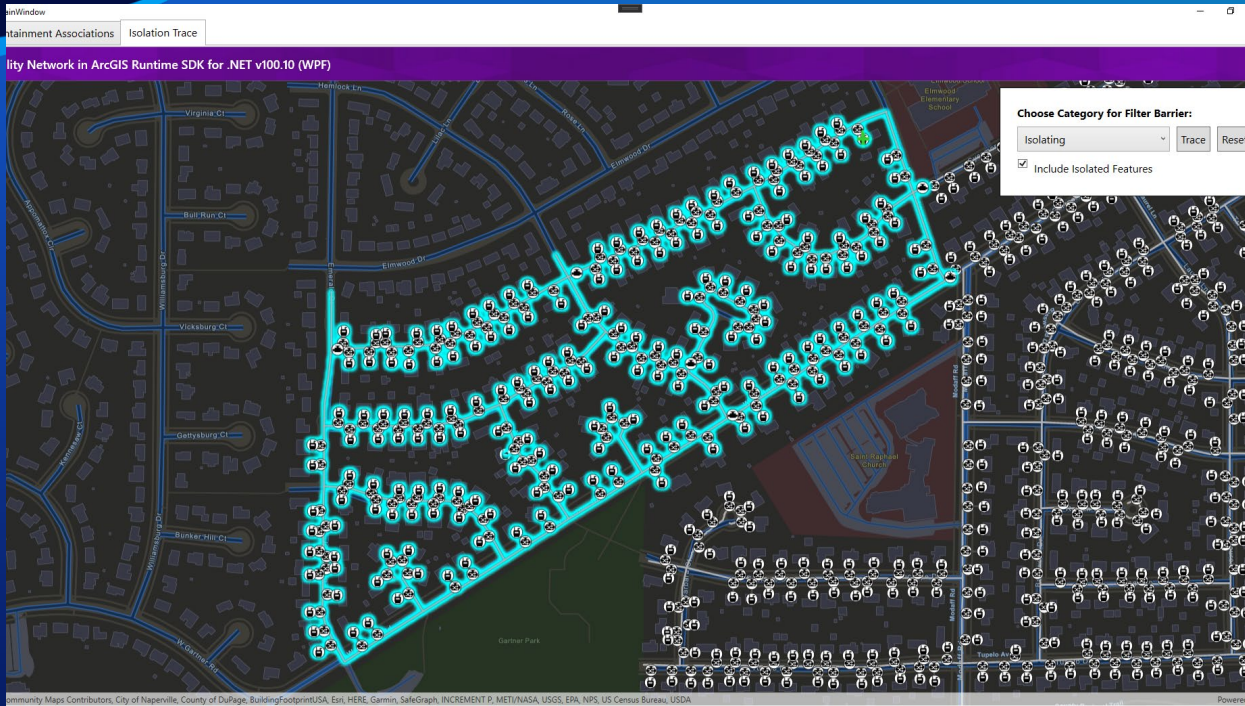
```
_parameters.TraceConfiguration.Functions.Add(traceFunction);
_parameters.ResultTypes.Add(UtilityTraceResultType.FunctionOutputs);
IEnumerable<UtilityTraceResult> traceResults = await _utilityNetwork.TraceAsync(_parameters);
foreach (UtilityTraceResult traceResult in traceResults)
{
    if(traceResult.Warnings.Count > 0)
    {
        MessageBox.Show(string.Join(Environment.NewLine, traceResult.Warnings));
    }
    if(traceResult is UtilityFunctionTraceResult functionTraceResult)
    {
        foreach(UtilityTraceFunctionOutput functionOutput in functionTraceResult.FunctionOutputs)
        {
            var result = functionOutput.Result; // for functionOutput.Function;
        }
    }
}
```

Input for shortest path trace

```
_parameters = new UtilityTraceParameters(UtilityTraceType.ShortestPath,  
                                          new UtilityElement[] { startingLocation });  
_parameters.Barriers.Add(barrierElement);  
_parameters.TraceConfiguration = new UtilityTraceConfiguration()  
{  
    ShortestPathNetworkAttribute =  
        _utilityNetwork.Definition.GetNetworkAttribute(networkAttributeName)  
};
```

Input for loops trace

```
_parameters = new UtilityTraceParameters(UtilityTraceType.Loops,  
                                          new UtilityElement[] { startingLocation });  
_parameters.Barriers.Add(barrierElement);
```

Isolation Trace

Jennifer Nery

Working with Utility Networks While Online

The background features a complex, abstract graphic design. It consists of various overlapping shapes and lines in shades of blue, red, and yellow. Some elements resemble stylized maps or network diagrams, while others are more fluid and organic. The overall aesthetic is modern and technical, fitting the theme of utility networks.

What Utility Network Capabilities are Available Online?

- Everything you've seen so far
 - Schema information
 - Associations
 - Tracing

What Utility Network Capabilities are Available Online?

- Everything you've seen so far
 - Schema information
 - Associations
 - Tracing
- And more!
 - Web map integration
 - Branch versioning

Web Map Integration

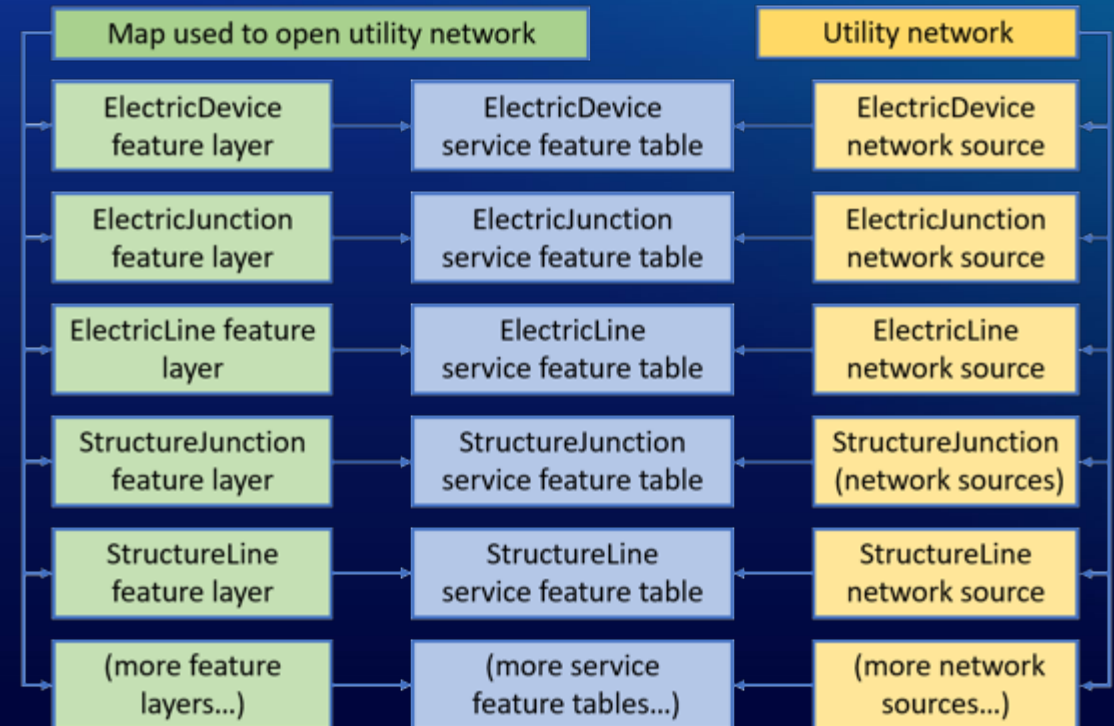
- If you publish a web map using ArcGIS Pro 2.7 to ArcGIS Enterprise 10.8.1...
... and include a utility network layer in your map
- The resulting map will have a utility network stored with it!
- **Map.UtilityNetworks**
 - New property at Runtime 100.10 that returns a list of utility networks in a map
- This is now the preferred way to create a **UtilityNetwork** object
 - Can still create a **UtilityNetwork** as before but this removes the need to hardcode the feature service URL in your application

Subtype Group Layers

- Subtype group layers can also be published by ArcGIS Pro and consumed by Runtime
 - Using Runtime's `SubtypeFeatureLayer` class
- To work correctly with the utility network, maps should be created with one layer for each network source
 - Typically, with one sublayer for each asset group
 - Allows changing symbology, zoom levels, even visible fields for each asset group

Why the One-layer Restriction?

- Association queries and trace results are returned as network source / GlobalID pairs
- Having one layer for each network source allows Runtime to map them back to features
- You don't need one layer for each network source... but you shouldn't have more than one



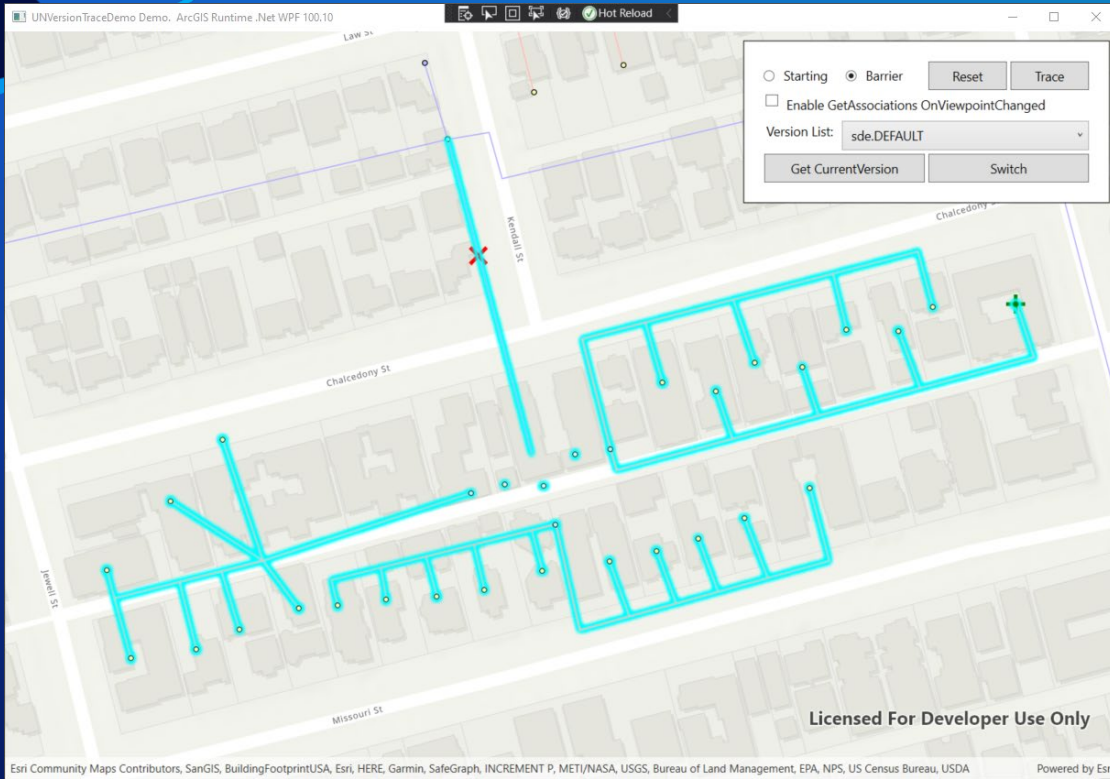
Versioning

- Versioning allows multiple editors to alter the same data without applying locks or duplicating data
- Feature services supports branch versioning
- Base version is called “Default”
- Other versions are created as children to Default
- Versioning concepts are detailed in other DevSummit presentations

- Branch versioning is not specific to utility networks!

The Service Geodatabase Class

- Introduced at Runtime 100.9
- Can be obtained using `UtilityNetwork.ServiceGeodatabase` or `ServiceFeatureTable.ServiceGeodatabase`
- `GetVersionsAsync` returns the available versions on this feature service
- `CreateVersionAsync` creates a new version
- `SwitchVersionAsync` switches the version for the tables on the service geodatabase
 - All utility network feature tables point to the same version
- Not supported- these are not common mobile workflows
 - Delete versions
 - Reconcile and post



Trace, working against a version

Tony Wakim

Get UtilityNetwork from Map (WebMap)

```
private async void LoadUNFromWebMap(object sender, RoutedEventArgs e)
{
    try
    {
        string UtilityNetworkWebMapUri = "<Provide WebMap URI here>";
        MyMapView.Map = new Map(new Uri(UtilityNetworkWebMapUri));
        await MyMapView.Map.LoadAsync();

        if (MyMapView.Map.UtilityNetworks.Count > 0)
        {
            _utilityNetwork = MyMapView.Map.UtilityNetworks[0];
            await _utilityNetwork.LoadAsync();
        }
        else
            MessageBox.Show("WebMap does not contain a UN!");

        // Get List of versions
        cboVersionList.Items.Clear();
        var versions = await _utilityNetwork.ServiceGeodatabase.GetVersionsAsync();
        foreach (var version in versions)
            cboVersionList.Items.Add(version.Name);
        cboVersionList.SelectedIndex = 0;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, ex.GetType().Name, MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
```


Define Starting Points and Barriers

private UtilityTraceParameters _parameters;

```
var result = await MyMapView.IdentifyLayersAsync(e.Position, 5, false);  
var feature = result.FirstOrDefault(r => r.GeoElements.Any(g => g is ArcGISFeature))?.GeoElements?.FirstOrDefault() as ArcGISFeature;  
if (feature == null)  
    return;
```

```
var element = _utilityNetwork.CreateElement(feature);
```

```
if (isStartingLocation)  
    _parameters.StartingLocations.Add(element);  
else  
    _parameters.Barriers.Add(element);
```

Trace

```
private async void OnTrace(object sender, RoutedEventArgs e)
{
    try
    {
        foreach (var layer in MyMapView.Map.OperationalLayers.OfType<FeatureLayer>())
        {
            layer.ClearSelection();
        }

        var result = await _utilityNetwork.TraceAsync(_parameters);

        if (result.FirstOrDefault() is UtilityElementTraceResult elementResult)
        {
            if (elementResult.Warnings.Count > 0)
                MessageBox.Show(string.Join("\n", elementResult.Warnings), "Trace Result Warnings", MessageBoxButton.OK);

            foreach (var layer in MyMapView.Map.OperationalLayers.OfType<FeatureLayer>())
            {
                var elements = elementResult.Elements.Where(e1 => e1.NetworkSource.FeatureTable == layer.FeatureTable);
                var features = await _utilityNetwork.GetFeaturesForElementsAsync(elements);
                layer.SelectFeatures(features);
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, ex.GetType().Name, MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
```

GetAssociations(extent)

```
private async Task SynthesizeAssociationGeometryAsync()
{
    try
    {
        var extent = MyMapView.GetCurrentViewpoint(ViewpointType.BoundingGeometry)?.TargetGeometry?.Extent;
        if (extent == null)
            return;


        var overlay = MyMapView.GraphicsOverlays[0];
        overlay.Graphics.Clear();

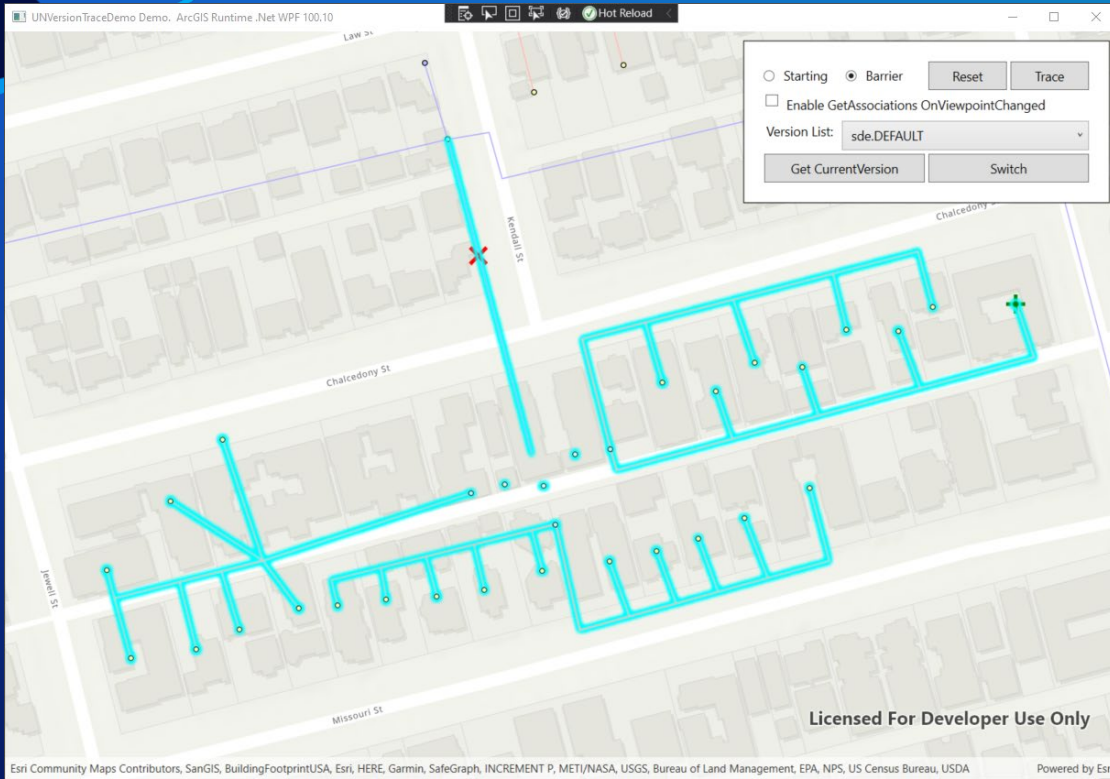
        var associations = await _utilityNetwork.GetAssociationsAsync(extent);

        foreach (var asso in associations)
        {
            var symbol = asso.AssociationType == UtilityAssociationType.Attachment ? AttachmentSymbol : ConnectivitySymbol;
            overlay.Graphics.Add(new Graphic(asso.Geometry, symbol));
        }
    }
    catch (TooManyAssociationsException) { }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, ex.GetType().Name, MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
```

Switch Version

```
private async void OnSwitch(object sender, RoutedEventArgs e)
{
    try
    {
        await _utilityNetwork.ServiceGeodatabase.SwitchVersionAsync(cboVersionList.SelectedItem.ToString());
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, ex.GetType().Name, MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
```





Trace, working against a version

Tony Wakim

Editing with Service Geodatabases

- Editing is fully supported
 - `ServiceGeodatabase.ApplyEditsAsync` / `ServiceGeodatabase.UndoLocalEdits()` can be used to save/revert all edits to a version using the same database transaction
- Supports multiple session types
 - `Transient` for typical mobile applications (the default)
 - `Persistent` for creating desktop applications (locks the version to provide better consistency)
- The `ServiceGeodatabase` class isn't specific to utility networks

Working with Utility Networks While Offline

The background features a complex, abstract graphic design. It consists of various overlapping shapes and lines in shades of blue, red, and yellow. Some elements resemble stylized maps or network diagrams, while others are more fluid and organic. The overall aesthetic is modern and technical, fitting the theme of utility networks.

Working with Utility Networks While Offline

- Utility network feature classes can be taken offline
 - Using sync (`OfflineMapSyncTask` and `GeodatabaseSyncTask`)
 - Using ArcGIS Pro and mobile map packages (read-only)
- Subtype group layers work with offline maps
- Utility network features can be edited*
 - Dirty areas will be created and rules will be run when the data is synced back to the service
- No support for utility network functionality... yet
 - Schema information
 - Associations
 - Tracing



The Road Ahead

Utility Network

Taking Utility Networks Offline

- `OfflineMapSyncTask` and `GeodatabaseSyncTask` will be enhanced to take core utility network information offline
- Phase One
 - ✓ Schema information
 - ✓ Associations
 - ✓ Map integration
 - Branch versioning (not applicable)
 - Tracing
- Editing
 - Same capabilities as before

Named Trace Configurations (Online)

- A Named trace configuration allows the data-model specific parameters of a trace to be treated as a “black box.”
- Created with ArcGIS Pro and published to the utility network service
 - Typically, by domain experts
- Included with web maps published by Pro
- Allows Runtime developers to write tracing code that is data model-independent
 - And even domain-independent



Display Filters

- Used to turn off the display of features based on a filter
- Unlike definition queries, does not restrict the feature from being returned by other queries
- Not utility network specific, but often used to hide features that are within containers

Limited Offline Editing

- Allow association editing
 - Supports field data corrections (ex: attaching a transformer to the correct pole)
 - Respecting rules that are configured with your network
- Respect association deletion semantics
 - E.g., delete a transformer bank, and the features contained within it are deleted as well

Offline Tracing

- Major cross-team development effort for 2021

Additional Editing Capabilities

- Attribute rules
 - Support for validation and batch calculation rules (online)
 - Support for immediate calculation and constraint rules (offline)
- Contingent validity (both online and offline)
- Validate topology (online first, eventually offline)

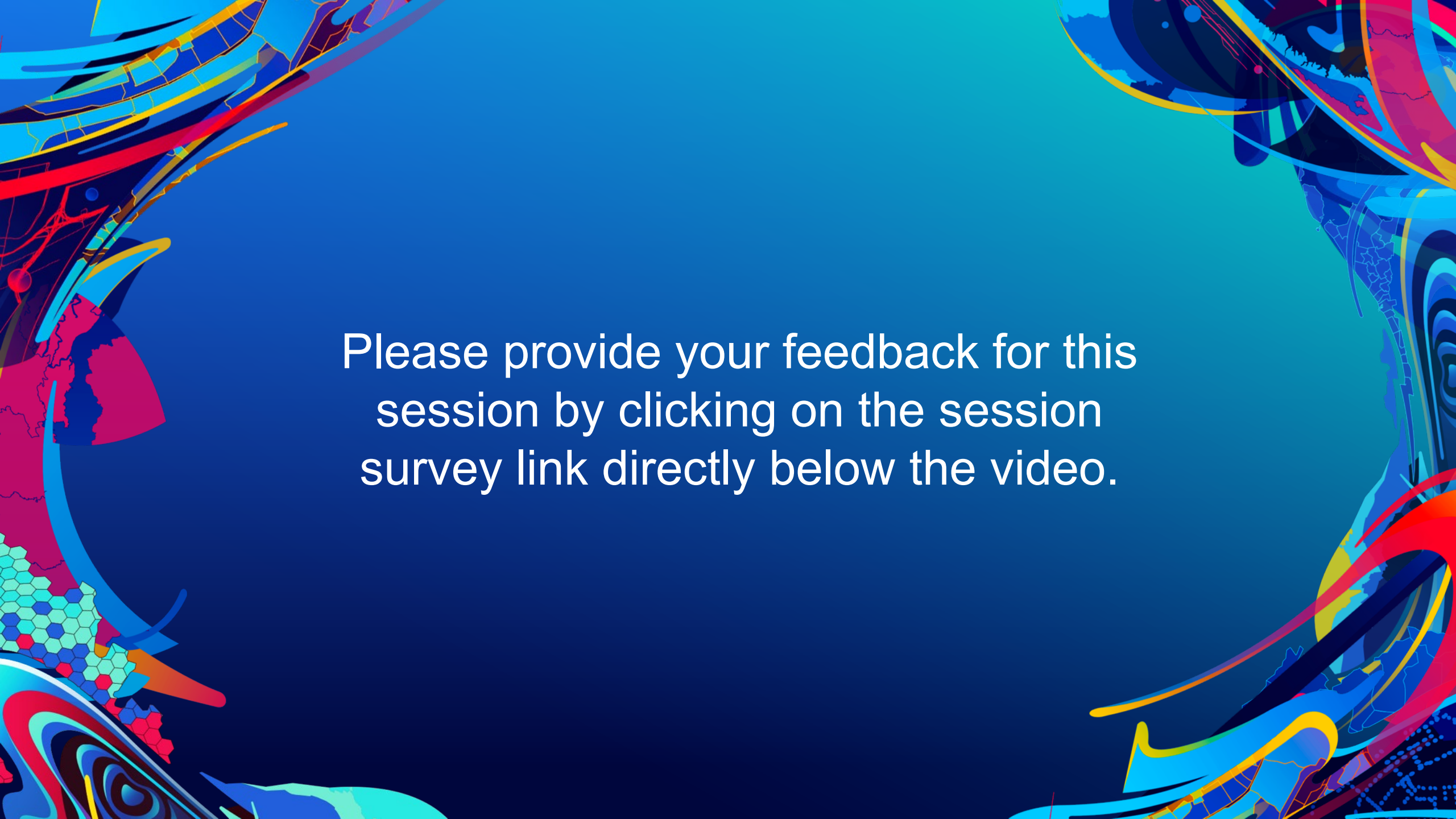
For Additional Information

- Additional DevSummit sessions
 - Network Management with ArcGIS: Deep Dive into the Capabilities of the Utility Network
 - Version Management with ArcGIS
 - ArcGIS Runtime SDK Building Apps sessions (multiple sessions and multiple platforms)
- Last year's session
 - Provides additional information about annotation and taking data offline
- Sample code from this session
- ArcGIS Runtime developer site



esri®

THE
SCIENCE
OF
WHERE®



Please provide your feedback for this session by clicking on the session survey link directly below the video.