2011 Esri Developer Summit
Palm Springs, CA

Geometric Networks for Developers

David Crawford

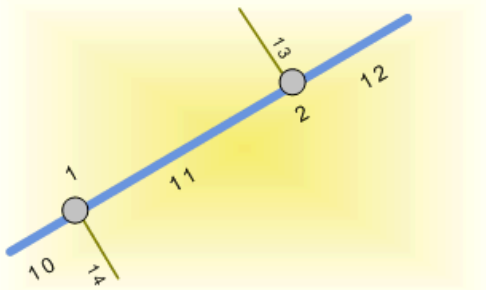
Alan Hatakeyama

Overview

- Brief review of geometric networks
- Creating geometric networks
- Adding connectivity rules
- Creating and modifying network features
- Performing analysis on a geometric network
- Creating a custom trace task
- Traversing a geometric network
- Questions

Geometric Networks

- Used to model network systems
  - Primarily designed for Utilities/Natural Resources industries
- Connectivity relationships between feature classes.
  - Can associate connectivity rules with the network.
  - Connectivity is based on geometric coincidence, always live.
  - Live within a Feature Dataset
- Each feature class has a role in the network
  - A network may have multiple feature classes in the same role.



Water mains (Lines)

OID	Shape	Diameter	Material
10		8	Concrete
11		10	PVC
12		8	Concrete

Water services (Lines)

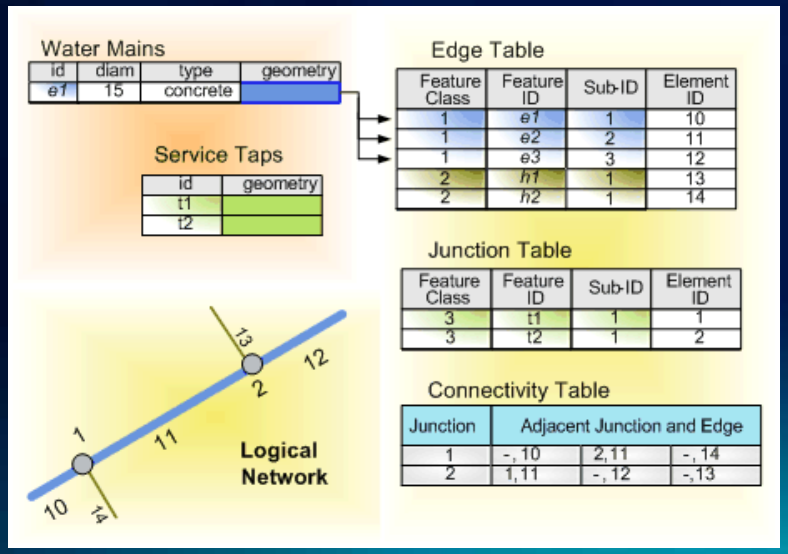
OID	Shape	Service ID	Material
13		1001	Cast iron
14		1002	Copper

Water junction fittings (Points)

OID	Shape	Equip ID	Valve Type
1		816-32	T203
2		816-45	Y53

Geometric Networks …

- A geometric network is associated with a logical network.
  - Each network feature is associated with one or more elements in the logical network.
  - Trace solvers on the logical network provide
- Connectivity tracing, cycle detection, flow directions
  - Upstream/downstream tracing, Isolation tracing



Downstream Trace

Creating Geometric Networks

How to create geometric networks within the geodatabase

- Use INetworkLoader for creation of geometric networks
  - Specify the input parameters for the geometric network
  - Use the LoadNetwork method to create the geometric network according to the specified parameters
- Parameters of note include:
  - Network name
  - Enabled and AncillaryRole field
  - Snapping and Snap tolerance
    - Uses the Tolerance for the Feature Dataset
  - Adding feature classes
    - Check if they are supported; INetworkLoader2::CanUseFeatureClass
  - Adding weights and weight associations
    - Fields must pre-exist
  - After building the network
    - Check for existence of build errors

Demo:
Create a geometric network

Connectivity Rules

- Allow you to constrain permissible connectivity
  - By default, any edge to any junction
- If any rule is specified, they must all be specified
  - Remember to include orphan junctions
- Edge-Junction rule
  - edge A may be connected to junction B
  - may have a default junction type (endpoint)
- Edge-Edge Rule
  - edge A may be connected to edge B via junction C
  - supports a default junction
  - edge-junction rules created as a side effect

Demo:
Add Connectivity Rules

Creating Network Features

Basic process to create feature

- CreateFeature
  - If subtypes present, set IRowSubtypes::SubtypeCode
- Call IRowSubtypes::InitDefaultValues
  - Enabled, Ancillary Roles will be handled
- Set attribute values
- Create geometry and set Shape
- Call Store
  - Writes the values to the record in the table

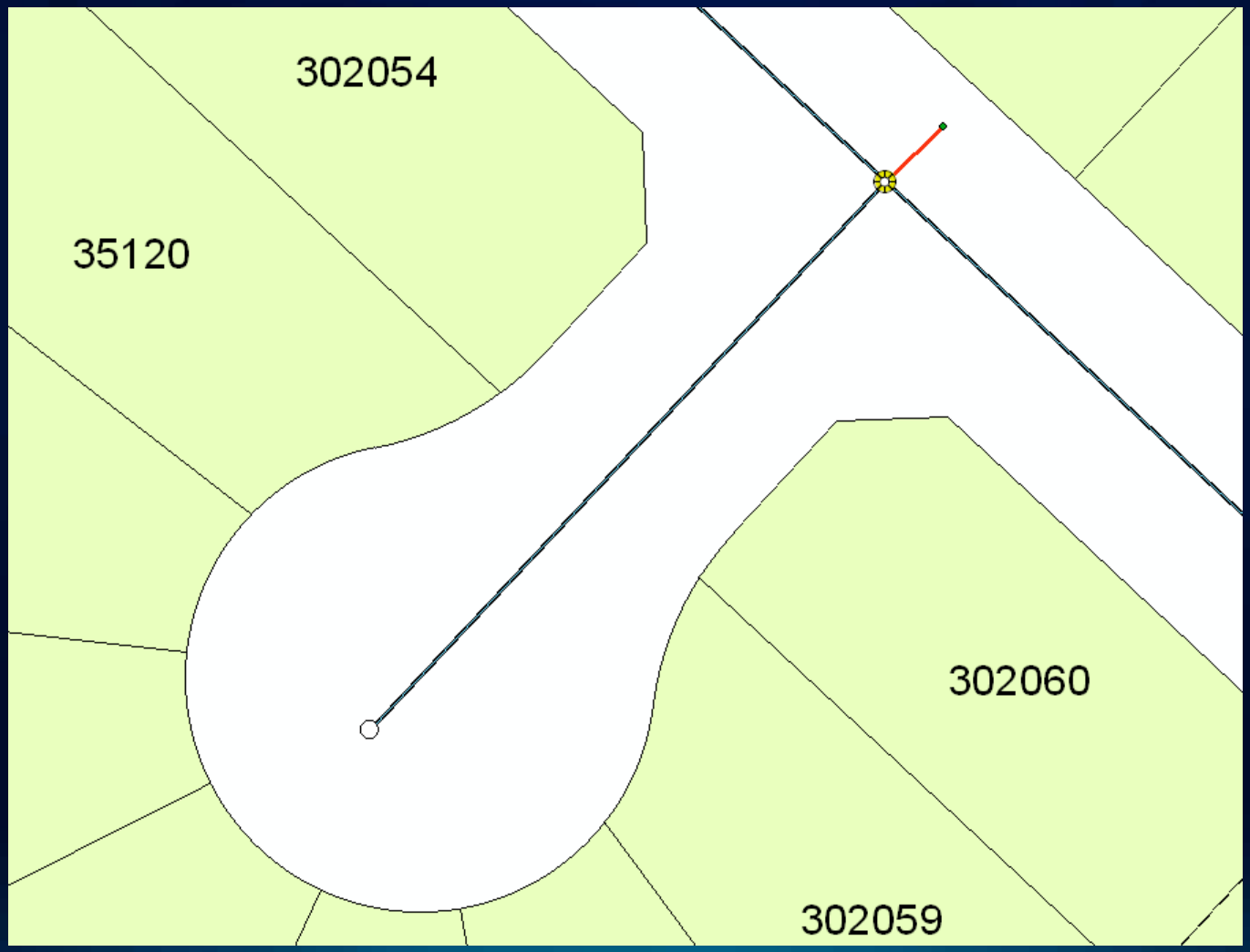
Creating Network Features …

- Geometric Network features are classified as complex features
  - Do not support non versioned edits
  - Must be edited with an Edit Session and Edit Operation
- Geometric Network specific behavior is handled by the Geometric Network at creation time
  - Not required to call Connect
  - Not required create any logical network connectivity; i.e.: CreateNetworkElements method
  - Enabled and AncillaryRole values are set by the feature
- Not required to call Disconnect and Connect with spatial updates to features; geometric network will ensure integrity
  - Unless, you want to edit the feature geometry without impacting connected features

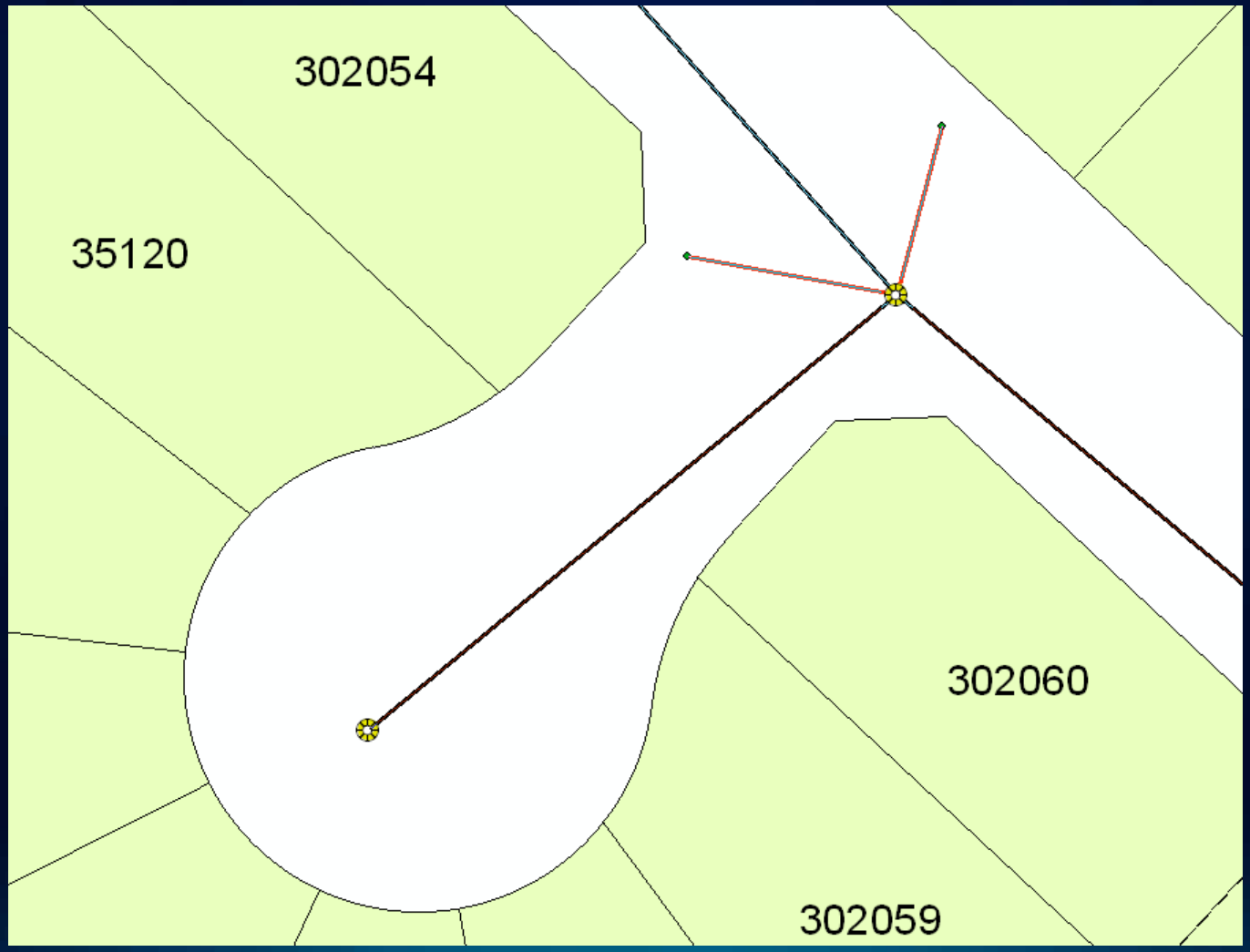
Creating Network Features …

- Cursors
  - Insert cursors can perform direct inserts outside of an edit session on simple data
    - Same rule applies to update cursors
    - Offers performance advantages; i.e.: events not fired
  - Using these APIs on network features negates any performance advantages
- Why?
  - All geometric network behavior is observed

Demo Outline



Demo Outline



Demo:
Create Network Features

Perform Analysis on a Geometric Network

The TraceFlowSolver object

- Performs basic analyses on a geometric network
  - Same analyses as trace tasks on Utility Network Analyst toolbar
- Inputs
  - Flags
  - Weights
  - Restrictions
- Returns the set of network elements traced

- Found in the esriNetworkAnalysis library

TraceFlowSolver object methods

Trace task on the Utility Network Analyst toolbar	Method on the TraceFlowSolver object
Find Common Ancestors	FindCommonAncestors()
Find Loops	FindCircuits()
Find Path	FindPath()
Find Path Upstream	FindSource()
Find Upstream Accumulation	FindAccumulation()
Find Disconnected	FindFlowUnreachedElements()
Find Connected Trace Downstream Trace Upstream	FindFlowElements() FindFlowEndElements()

Performing analysis on a geometric network

- 1. Setting up the TraceFlowSolver object
- 2. Specifying flags
- 3. Solving an analysis
- 4. Extracting the results

1. Setting up the TraceFlowSolver object

```
'Create the TraceFlowSolver object
Dim tfs As ITraceFlowSolverGEN = _
   CType(New TraceFlowSolver(), ITraceFlowSolverGEN)

'Specify the network to analyze
Dim netSolver As INetSolver = CType(tfs, INetSolver)
netSolver.SourceNetwork = geomNet.Network

'...
```

1. Setting up the TraceFlowSolver object (cont'd)

```
'Specify the weights to use
Dim solverWeights As INetSolverWeightsGEN = _
              CType(tfs, INetSolverWeightsGEN)
Dim netSchema As INetSchema = _
              CType(geomNet.Network, INetSchema)
solverWeights.FromToEdgeWeight = _
                   netSchema.WeightByName("Length")
solverWeights.ToFromEdgeWeight = _
                   netSchema.WeightByName("Length")

'Specify any restrictions
netSolver.DisableElementClass(restrFC.FeatureClassID)
tfs.TraceIndeterminateFlow = False
'...etc.
```

2. Specifying flags

```
'Create and populate an EdgeFlag object
Dim netFlag As INetFlag = CType(New EdgeFlag(), INetFlag)
netFlag.UserClassID = fc.FeatureClassID
netFlag.UserID = feature.OID
netFlag.SubID = 0
Dim pEdgeFlag As IEdgeFlag = CType(netFlag, IEdgeFlag)
edgeFlag.Position = 0.5

'Pass the flag as an array to TraceFlowSolver object
Dim edgeFlagArray(0 To 0) As IEdgeFlag
edgeFlagArray(0) = edgeFlag
tfs.PutEdgeOrigins(edgeFlagArray)
```

The SubID

- Determines the specific network element of given feature
- SubID = 0 for junction features and simple edge features
- SubID >= 0 for complex edge features

- SubID values do NOT necessarily correspond to the ordering of edge elements within the feature
- SubID values are NOT necessarily consecutive

Determining the SubID for a ComplexEdgeFeature

- Look up the EID
  - IComplexNetworkFeature::FindEdgeEID(), or
  - Use the PointToEID object

- Convert the EID to ClassID/ID/SubID triplet:

```
Dim netElements As INetElements = _
      CType(geomNet.Network, INetElements)
netElements.QueryIDs(inputEID, esriETEdge, _
      outputUserClassID, outputUserID, outputUserSubID)
```

3. Solving an analysis

```
'Create result enumerations
Dim resultJunctions As IEnumNetEID = _
                       New EnumNetEIDArray()
Dim resultEdges As IEnumNetEID = _
                       New EnumNetEIDArray()
Dim totalCost As Variant

'Perform an analysis
tfs.FindAccumulation(esriFMDownstream, _
                     esriFEJunctionsAndEdges, _
                     resultJunctions, resultEdges, _
                     totalCost)
```

The EIDHelper object

- Looks up features and/or geometries from an enumeration of EIDs
- Geometries are returned in the specified OutputSpatialReference
- Can return only those features/geometries within the given Envelope
  - IEIDHelper::putref_DisplayEnvelope()
- Returns features with only those field values of interest
  - IEIDHelper::AddField()

4. Extracting the results

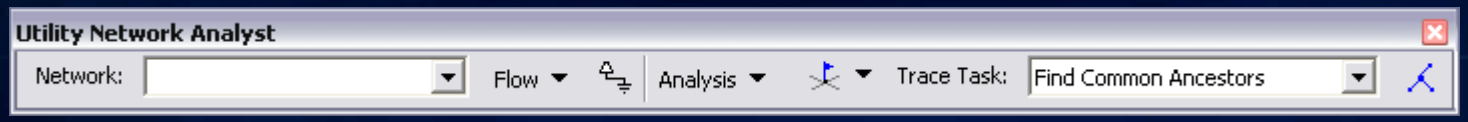
```
'Setup an EIDHelper object
Dim eidHelper As IEIDHelper = New EIDHelper()
eidHelper.GeometricNetwork = geomNet
eidHelper.OutputSpatialReference = sr
eidHelper.ReturnFeatures = True
eidHelper.ReturnGeometries = False
eidHelper.AddField("LinearRef_ID")

'...
```

4. Extracting the results (cont'd)

```
'Enumerate features in the results
Dim enumEIDInfo As IEnumEIDInfo = _
            eidHelper.CreateEnumEIDInfo(resultEdges)
enumEIDInfo.Reset()
Dim eidInfo As IEIDInfo = pEnumEIDInfo.Next()
Do Until eidInfo Is Nothing
  Dim eidInfoFeature As IFeature = _
            eidInfoFeature = eidInfo.Feature
  Console.WriteLine( _
          eidInfoFeature.Value(iLinearRefFieldIndex))
  eidInfo = enumEIDInfo.Next()
Loop
```

Creating a custom trace task



Creating a custom trace task

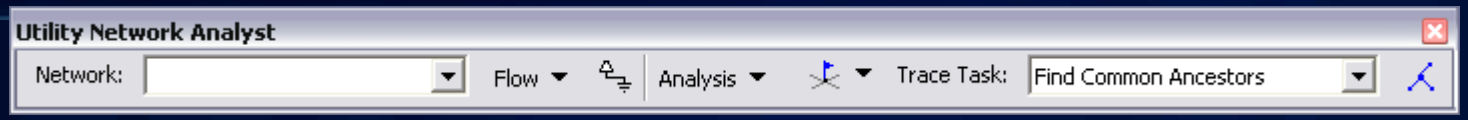
- Create a COM Class that implements:
  - ITraceTask and
  - ITraceTaskResults

- During COM Class registration/unregistration, register/unregister it as one of the UtilityNetworkTasks:
  - ESRI.ArcGIS.ADF.CATIDs.UtilityNetworkTasks.Register()
  - ESRI.ArcGIS.ADF.CATIDs.UtilityNetworkTasks.Unregister()

Methods to implement

- ITraceTask::OnCreate()
  - Logic executed when the trace task is loaded into ArcMap
- ITraceTask::get_Name()
  - The name of the trace task as displayed in the Utility Network Analyst toolbar
- ITraceTask::get_EnableSolve()
  - Logic determining when the Solve button should be enabled
  - Frequently executed – should be lightweight code
- ITraceTask::OnTraceExecution()
  - Logic executed when the Solve button is pressed
- ITraceTaskResults::get_Result{Edges/Junctions}()
  - Enumeration of network elements in the result set

Accessing the Utility Network Analyst toolbar GUI



- All settings on the Utility Network Analyst toolbar can be accessed from the UtilityNetworkAnalysisExt object
  - Useful for transferring user's settings to TraceFlowSolver object
- Found in the esriEditorExt library
- A reference to the UtilityNetworkAnalysisExt object is passed in when calling ITraceTask::OnCreate()

Sample Trace Task:
“Custom Upstream Trace Task”

Traversing a geometric network

The ForwardStar object

- Given a network element, returns all adjacent network elements and their weight values
- Create by calling INetwork::CreateForwardStar()
  - Specify honorState = True to only return non-Disabled elements
- Usage:
  - First call FindAdjacent() to determine the # of adjacent elements
  - Then call the Query…() methods to fetch each adjacent element and its weight value
- Found in the esriGeoDatabase library

ForwardStar example

```
'Get the network weights
Dim netSchema As INetSchema = _
            CType(geomNet.Network, INetSchema)
Dim juncWeight As INetWeight = _
            netSchema.WeightByName("JuncImpedance")
Dim ftEdgeWeight As INetWeight = _
            netSchema.WeightByName("FTEdgeImpedance")
Dim tfEdgeWeight As INetWeight = _
            netSchema.WeightByName("TFEdgeImpedance")

'Create ForwardStar object
Dim fs As IForwardStarGEN = _
         geomNet.Network.CreateForwardStar(True, _
         juncWeight, ftEdgeWeight, tfEdgeWeight, Nothing)
'...
```

ForwardStar example (cont'd)

```
Do Until theEntireNetworkIsTraversed
  'First determine number of adjacencies
  Dim numAdjacencies As Integer
  fs.FindAdjacent(incomingEdgeEID, _
                  currentJunctionEID, numAdjacencies)

  'Then loop through the adjacent elements
  For i As Integer = 0 To numAdjacencies - 1
    fs.QueryAdjacentEdge(i, adjEdgeEID, _
              adjEdgeOrientation, adjEdgeWeight)
    fs.QueryAdjacentJunction(i, adjJunctionEID, _
                             adjJunctionWeight)
    '...Do Something With Adjacency Information...
  Next i
Loop
```

Questions?

esri