```
void Main()
{
	// Create a List of EventRows ( This is the Main Document )
	List<EventRow> EventRowList = new List<EventRow>();
	// Create New EventRow ( Current EventRow )
	EventRow CurrentEventRow = new EventRow();
	// Create New EventRow ( Previous EventRow )
	EventRow PreviousEventRow = new EventRow();

	// Read a Kml file into List of Placemarks
	List<Placemark> Placemarks = GetPlaceMarkList("Brenco Plant Geofence.kml");
	// Create List of all the current Mac Address and NodeIDs
	List<AssetMap> AssetMapList = GetAssetMapping();

	// List of Current Device's Events
	List<Event> DeviceEventList = new List<Event>();

	// Loop Through Device, then all the Device's Events
	foreach (AssetMap am in AssetMapList.Take(1))
	{
		// Get All Events that include our device
		DeviceEventList.Clear();
		DeviceEventList = Events
		.Where( e => e.AdditionalProperties
			.Any(ap =>
			(ap.AdditionalPropertyAttributeID == 30 && ap.Value ==
am.Mac.ToString())
			|| (ap.AdditionalPropertyAttributeID == 37 && ap.Value ==
am.Mac.ToString())
			|| (ap.AdditionalPropertyAttributeID == 38 && ap.Value ==
am.Mac.ToString())
			//|| (ap.AdditionalPropertyAttributeID == 7) //This is to get
NetworkInfo Events
			))
		.Where (e => e.EventDate >= DateTime.Parse("03/07/2016"))
		.OrderBy (e => e.EventDate)
		//.Take(1)
		.ToList();

		foreach (Event thisEvent in DeviceEventList)
		{
			// Add MacAdress, EventTime, and EventName
			CurrentEventRow.MacAddress = am.Mac;

			if (staticDevices.Contains(am.Mac)) CurrentEventRow.DeviceType =
"Static";
			else CurrentEventRow.DeviceType = "Mobile";

			CurrentEventRow.EventTime = (DateTime)thisEvent.EventDate;
			CurrentEventRow.EventName = thisEvent.Name;

			// If ( Event.Name.Contains Location )
			#region Location
			if (thisEvent.Name.Contains("Location"))
			{
```

```
                        // Add Lat Long
                        CurrentEventRow.Latitude =
thisEvent.Locations.FirstOrDefault().Latitude;
                        CurrentEventRow.Longitude =
thisEvent.Locations.FirstOrDefault().Longitude;
                        // Set LocationUpdateTime
                        CurrentEventRow.LocationUpTime
=(DateTime)thisEvent.EventDate;
                  }
                  // Else If ( Previous EventRow Exists && has lat long )
                  else if (PreviousEventRow != null && PreviousEventRow.Latitude !=
null)
                  {
                        // Previous Lat Long = Current Lat Long
                        CurrentEventRow.Latitude = PreviousEventRow.Latitude;
                        CurrentEventRow.Longitude = PreviousEventRow.Longitude;
                        // Keep old LocationUpdateTime
                        CurrentEventRow.LocationUpTime =
(DateTime)PreviousEventRow.LocationUpTime;
                  }
                  // Else
                  else
                  {
                        // Leave Lat Long Empty
                        CurrentEventRow.Latitude = null;
                        CurrentEventRow.Longitude = null;
                  }
                  #endregion

                  // Add PlaceMarks from Lat & Long ( if exist )
                  #region Placemarks
                  if (CurrentEventRow.Latitude != null)
                  {
                        decimal? x = CurrentEventRow.Latitude;
                        decimal? y = CurrentEventRow.Longitude;
                        bool HasFirstPlacemark = false;

                        foreach (Placemark placemark in Placemarks)
                        {
                              if(
placemark.Geometry.CalculateBounds().Contains(Double.Parse(x.ToString()),
Double.Parse(y.ToString())))
                              {
                                    if(HasFirstPlacemark)
                                    {
                                          CurrentEventRow.PlaceMark2 =
placemark.Name;
                                    }
                                    else
                                    {
                                          CurrentEventRow.PlaceMark1 =
placemark.Name;
                                    }
                              }
                        }
```

```
                    }
                    #endregion

                    // If ( Event.Name.Contains Motion )
                    #region Motion
                    if (thisEvent.Name.Contains("Motion"))
                    {
                          // If Dwell = False, If Moving = True
                          if (thisEvent.AdditionalProperties.Where(ap => ap.Value ==
"Moving").Any())
                          {
                                CurrentEventRow.IsMoving = true;
                          }
                          if (thisEvent.AdditionalProperties.Where(ap => ap.Value !=
"Moving").Any())
                          {
                                CurrentEventRow.IsMoving = false;
                          }
                    }
                    // Else if ( Previous EventRow Exists && has IsMoving)
                    else if (PreviousEventRow.IsMoving)
                    {
                          CurrentEventRow.IsMoving = true;
                    }
                    // Else ( keep it that way )
                    else
                    {
                          // Leave IsMoving Empty
                          CurrentEventRow.IsMoving = false;
                    }
                    #endregion

                    // If ( Event.Name Contains Path )
                    #region Network Path
                    if (thisEvent.Name.Contains("Path"))
                    {
                          // Add PathSource, PathDestination, Direction
                          CurrentEventRow.PathSource =
thisEvent.AdditionalProperties.Where(ap => ap.AdditionalPropertyAttributeID ==
37).First().Value;
                          CurrentEventRow.PathDestination =
thisEvent.AdditionalProperties.Where(ap => ap.AdditionalPropertyAttributeID ==
38).First().Value;
                          CurrentEventRow.Direction =
thisEvent.AdditionalProperties.Where(ap => ap.AdditionalPropertyAttributeID ==
39).First().Value;
                    }
                    #endregion

                    // If ( Event.Name Contains NetworkInfo )
                    #region Network Information
//                  if (thisEvent.Name.Contains("Network Information"))
//                  {
//                        // Add Network Info Columns
```

```
//                        CurrentEventRow.NumMotes =
int.Parse(thisEvent.AdditionalProperties.Where(ap => ap.AdditionalPropertyAttributeID
== 7).First().Value);
//                        CurrentEventRow.AsnSize =
int.Parse(thisEvent.AdditionalProperties.Where(ap => ap.AdditionalPropertyAttributeID
== 8).First().Value);
//                        CurrentEventRow.NetReliability =
int.Parse(thisEvent.AdditionalProperties.Where(ap => ap.AdditionalPropertyAttributeID
== 11).First().Value);
//                        CurrentEventRow.NetPathStability =
int.Parse(thisEvent.AdditionalProperties.Where(ap => ap.AdditionalPropertyAttributeID
== 12).First().Value);
//                        CurrentEventRow.NetLatency =
int.Parse(thisEvent.AdditionalProperties.Where(ap => ap.AdditionalPropertyAttributeID
== 13).First().Value);
//                  }
//                  else
//                  {
//                        // Add Network Info Columns from Previous
//                        CurrentEventRow.NumMotes = PreviousEventRow.NumMotes;
//                        CurrentEventRow.AsnSize = PreviousEventRow.AsnSize;
//                        CurrentEventRow.NetReliability =
PreviousEventRow.NetReliability;
//                        CurrentEventRow.NetPathStability =
PreviousEventRow.NetPathStability;
//                        CurrentEventRow.NetLatency = PreviousEventRow.NetLatency;
//                  }
                  #endregion

                  // If ( Event.Name Contains Device Health Info )
                  #region Device Health
                  if (thisEvent.Name.Contains("Device Health"))
                  {
                        // Add Device Health Info Columns
                        CurrentEventRow.DeviceHealthUpTime = thisEvent.EventDate;
                        CurrentEventRow.Charge =
int.Parse(thisEvent.AdditionalProperties.Where(ap => ap.AdditionalPropertyAttributeID
== 17).First().Value);
                        CurrentEventRow.Temperature =
int.Parse(thisEvent.AdditionalProperties.Where(ap => ap.AdditionalPropertyAttributeID
== 19).First().Value);
                        CurrentEventRow.Voltage =
int.Parse(thisEvent.AdditionalProperties.Where(ap => ap.AdditionalPropertyAttributeID
== 20).First().Value);
                        CurrentEventRow.NumTxOk =
int.Parse(thisEvent.AdditionalProperties.Where(ap => ap.AdditionalPropertyAttributeID
== 21).First().Value);
                        CurrentEventRow.NumTxFailure =
int.Parse(thisEvent.AdditionalProperties.Where(ap => ap.AdditionalPropertyAttributeID
== 22).First().Value);
                        CurrentEventRow.NumRxOk =
int.Parse(thisEvent.AdditionalProperties.Where(ap => ap.AdditionalPropertyAttributeID
== 23).First().Value);
```

```
                        CurrentEventRow.NumRxFailure =
int.Parse(thisEvent.AdditionalProperties.Where(ap => ap.AdditionalPropertyAttributeID
== 24).First().Value);
                    }
                    else
                    {
                        // Add Previous Device Health Info Columns
                        CurrentEventRow.DeviceHealthUpTime =
PreviousEventRow.DeviceHealthUpTime;
                        CurrentEventRow.Charge = PreviousEventRow.Charge;
                        CurrentEventRow.Temperature =
PreviousEventRow.Temperature;
                        CurrentEventRow.Voltage = PreviousEventRow.Voltage;
                        CurrentEventRow.NumTxOk = PreviousEventRow.NumTxOk;
                        CurrentEventRow.NumTxFailure =
PreviousEventRow.NumTxFailure;
                        CurrentEventRow.NumRxOk = PreviousEventRow.NumRxOk;
                        CurrentEventRow.NumRxFailure =
PreviousEventRow.NumRxFailure;
                    }
                    #endregion

                    // If ( Event.Name Contains Neighbor Health Info )
                    #region Neighbor Health
                    if (thisEvent.Name.Contains("Neighbor Health"))
                    {
                        //Update Time
                        CurrentEventRow.NeighorHealthUpTime = thisEvent.EventDate;

                        // Add Neighbor Health Info Columns
                        String jsonString = thisEvent.AdditionalProperties.Where
(ap => ap.AdditionalPropertyAttributeID == 32).First().Value;
                        jsonString = "[" + jsonString + "]";
                        List<NeighborStatus> nslist =
JsonConvert.DeserializeObject<List<NeighborStatus>>(jsonString);
                        //nslist.Dump(CurrentEventRow.MacAddress);

                        //Catch Invalid NeighborID's
                        // TODO - Make this Dynamic with the AssetMapping List
                        if (nslist.Where (n => n.NeighborID > 31).Any())
                        {
                            nslist.Remove(nslist.Where(n => n.NeighborID >
31).First());
                        }

                        int counter = 0;

                        foreach(NeighborStatus status in nslist)
                        {
                            counter++;
                            if(counter == 1)
                            {
                                CurrentEventRow.N1MacAddress =
AssetMapList.Where(a => a.NodeID == status.NeighborID).First().Mac;
                                CurrentEventRow.N1RSI = status.RRSI;
```

```
                                        CurrentEventRow.N1NumTxPackets =
status.NumTxPackets;
                                        CurrentEventRow.N1NumTxFailures =
status.NumTxFailures;
                                        CurrentEventRow.N1NumRxPackets =
status.NumRxPackets;
                                }

                                if(counter == 2)
                                {
                                        CurrentEventRow.N2MacAddress =
AssetMapList.Where(a => a.NodeID == status.NeighborID).First().Mac;
                                        CurrentEventRow.N2RSI = status.RRSI;
                                        CurrentEventRow.N2NumTxPackets =
status.NumTxPackets;
                                        CurrentEventRow.N2NumTxFailures =
status.NumTxFailures;
                                        CurrentEventRow.N2NumRxPackets =
status.NumRxPackets;
                                }

                                if(counter == 3)
                                {
                                        CurrentEventRow.N3MacAddress =
AssetMapList.Where(a => a.NodeID == status.NeighborID).First().Mac;
                                        CurrentEventRow.N3RSI = status.RRSI;
                                        CurrentEventRow.N3NumTxPackets =
status.NumTxPackets;
                                        CurrentEventRow.N3NumTxFailures =
status.NumTxFailures;
                                        CurrentEventRow.N3NumRxPackets =
status.NumRxPackets;
                                }

                                if(counter == 4)
                                {
                                        CurrentEventRow.N4MacAddress =
AssetMapList.Where(a => a.NodeID == status.NeighborID).First().Mac;
                                        CurrentEventRow.N4RSI = status.RRSI;
                                        CurrentEventRow.N4NumTxPackets =
status.NumTxPackets;
                                        CurrentEventRow.N4NumTxFailures =
status.NumTxFailures;
                                        CurrentEventRow.N4NumRxPackets =
status.NumRxPackets;
                                }

                                if(counter == 5)
                                {
                                        CurrentEventRow.N5MacAddress =
AssetMapList.Where(a => a.NodeID == status.NeighborID).First().Mac;
                                        CurrentEventRow.N5RSI = status.RRSI;
                                        CurrentEventRow.N5NumTxPackets =
status.NumTxPackets;
```

```
                                        CurrentEventRow.N5NumTxFailures =
status.NumTxFailures;
                                        CurrentEventRow.N5NumRxPackets =
status.NumRxPackets;
                                }

                                if(counter == 6)
                                {
                                        CurrentEventRow.N6MacAddress =
AssetMapList.Where(a => a.NodeID == status.NeighborID).First().Mac;
                                        CurrentEventRow.N6RSI = status.RRSI;
                                        CurrentEventRow.N6NumTxPackets =
status.NumTxPackets;
                                        CurrentEventRow.N6NumTxFailures =
status.NumTxFailures;
                                        CurrentEventRow.N6NumRxPackets =
status.NumRxPackets;
                                }

                                if(counter == 7)
                                {
                                        CurrentEventRow.N7MacAddress =
AssetMapList.Where(a => a.NodeID == status.NeighborID).First().Mac;
                                        CurrentEventRow.N7RSI = status.RRSI;
                                        CurrentEventRow.N7NumTxPackets =
status.NumTxPackets;
                                        CurrentEventRow.N7NumTxFailures =
status.NumTxFailures;
                                        CurrentEventRow.N7NumRxPackets =
status.NumRxPackets;
                                }

                                if(counter == 8)
                                {
                                        CurrentEventRow.N8MacAddress =
AssetMapList.Where(a => a.NodeID == status.NeighborID).First().Mac;
                                        CurrentEventRow.N8RSI = status.RRSI;
                                        CurrentEventRow.N8NumTxPackets =
status.NumTxPackets;
                                        CurrentEventRow.N8NumTxFailures =
status.NumTxFailures;
                                        CurrentEventRow.N8NumRxPackets =
status.NumRxPackets;
                                }
                        }

                        if (nslist.Count < 1)
                        {
                                CurrentEventRow.N1MacAddress = null;
                                CurrentEventRow.N1RSI = null;
                                CurrentEventRow.N1NumTxPackets = null;
                                CurrentEventRow.N1NumTxFailures = null;
                                CurrentEventRow.N1NumRxPackets = null;
                        }
                        if (nslist.Count < 2)
```

```
{
        CurrentEventRow.N2MacAddress = null;
        CurrentEventRow.N2RSI = null;
        CurrentEventRow.N2NumTxPackets = null;
        CurrentEventRow.N2NumTxFailures = null;
        CurrentEventRow.N2NumRxPackets = null;
}

if (nslist.Count < 2)
{
        CurrentEventRow.N2MacAddress = null;
        CurrentEventRow.N2RSI = null;
        CurrentEventRow.N2NumTxPackets = null;
        CurrentEventRow.N2NumTxFailures = null;
        CurrentEventRow.N2NumRxPackets = null;
}

if (nslist.Count < 3)
{
        CurrentEventRow.N3MacAddress = null;
        CurrentEventRow.N3RSI = null;
        CurrentEventRow.N3NumTxPackets = null;
        CurrentEventRow.N3NumTxFailures = null;
        CurrentEventRow.N3NumRxPackets = null;
}

if (nslist.Count < 4)
{
        CurrentEventRow.N4MacAddress = null;
        CurrentEventRow.N4RSI = null;
        CurrentEventRow.N4NumTxPackets = null;
        CurrentEventRow.N4NumTxFailures = null;
        CurrentEventRow.N4NumRxPackets = null;
}

if (nslist.Count < 5)
{
        CurrentEventRow.N5MacAddress = null;
        CurrentEventRow.N5RSI = null;
        CurrentEventRow.N5NumTxPackets = null;
        CurrentEventRow.N5NumTxFailures = null;
        CurrentEventRow.N5NumRxPackets = null;
}

if (nslist.Count < 6)
{
        CurrentEventRow.N6MacAddress = null;
        CurrentEventRow.N6RSI = null;
        CurrentEventRow.N6NumTxPackets = null;
        CurrentEventRow.N6NumTxFailures = null;
        CurrentEventRow.N6NumRxPackets = null;
}

if (nslist.Count < 7)
{
```

```
                                CurrentEventRow.N7MacAddress = null;
                                CurrentEventRow.N7RSI = null;
                                CurrentEventRow.N7NumTxPackets = null;
                                CurrentEventRow.N7NumTxFailures = null;
                                CurrentEventRow.N7NumRxPackets = null;
                        }

                        if (nslist.Count < 8)
                        {
                                CurrentEventRow.N8MacAddress = null;
                                CurrentEventRow.N8RSI = null;
                                CurrentEventRow.N8NumTxPackets = null;
                                CurrentEventRow.N8NumTxFailures = null;
                                CurrentEventRow.N8NumRxPackets = null;
                        }
                }
                else
                {
                #region Keep Neighbor Health Info
                                CurrentEventRow.N1MacAddress =
PreviousEventRow.N1MacAddress;
                                CurrentEventRow.N1RSI = PreviousEventRow.N1RSI;
                                CurrentEventRow.N1NumTxPackets =
PreviousEventRow.N1NumTxPackets;
                                CurrentEventRow.N1NumTxFailures =
PreviousEventRow.N1NumTxFailures;
                                CurrentEventRow.N1NumRxPackets =
PreviousEventRow.N1NumRxPackets;
                                CurrentEventRow.N2MacAddress =
PreviousEventRow.N2MacAddress;
                                CurrentEventRow.N2RSI = PreviousEventRow.N2RSI;
                                CurrentEventRow.N2NumTxPackets =
PreviousEventRow.N2NumTxPackets;
                                CurrentEventRow.N2NumTxFailures =
PreviousEventRow.N2NumTxFailures;
                                CurrentEventRow.N2NumRxPackets =
PreviousEventRow.N2NumRxPackets;
                                CurrentEventRow.N3MacAddress =
PreviousEventRow.N3MacAddress;
                                CurrentEventRow.N3RSI = PreviousEventRow.N3RSI;
                                CurrentEventRow.N3NumTxPackets =
PreviousEventRow.N3NumTxPackets;
                                CurrentEventRow.N3NumTxFailures =
PreviousEventRow.N3NumTxFailures;
                                CurrentEventRow.N3NumRxPackets =
PreviousEventRow.N3NumRxPackets;
                                CurrentEventRow.N4MacAddress =
PreviousEventRow.N4MacAddress;
                                CurrentEventRow.N4RSI = PreviousEventRow.N4RSI;
                                CurrentEventRow.N4NumTxPackets =
PreviousEventRow.N4NumTxPackets;
                                CurrentEventRow.N4NumTxFailures =
PreviousEventRow.N4NumTxFailures;
                                CurrentEventRow.N4NumRxPackets =
PreviousEventRow.N4NumRxPackets;
```

```
                                CurrentEventRow.N5MacAddress =
PreviousEventRow.N5MacAddress;
                                CurrentEventRow.N5RSI = PreviousEventRow.N5RSI;
                                CurrentEventRow.N5NumTxPackets =
PreviousEventRow.N5NumTxPackets;
                                CurrentEventRow.N5NumTxFailures =
PreviousEventRow.N5NumTxFailures;
                                CurrentEventRow.N5NumRxPackets =
PreviousEventRow.N5NumRxPackets;
                                CurrentEventRow.N6MacAddress =
PreviousEventRow.N6MacAddress;
                                CurrentEventRow.N6RSI = PreviousEventRow.N6RSI;
                                CurrentEventRow.N6NumTxPackets =
PreviousEventRow.N6NumTxPackets;
                                CurrentEventRow.N6NumTxFailures =
PreviousEventRow.N6NumTxFailures;
                                CurrentEventRow.N6NumRxPackets =
PreviousEventRow.N6NumRxPackets;
                                CurrentEventRow.N7MacAddress =
PreviousEventRow.N7MacAddress;
                                CurrentEventRow.N7RSI = PreviousEventRow.N7RSI;
                                CurrentEventRow.N7NumTxPackets =
PreviousEventRow.N7NumTxPackets;
                                CurrentEventRow.N7NumTxFailures =
PreviousEventRow.N7NumTxFailures;
                                CurrentEventRow.N7NumRxPackets =
PreviousEventRow.N7NumRxPackets;
                                CurrentEventRow.N8MacAddress =
PreviousEventRow.N8MacAddress;
                                CurrentEventRow.N8RSI = PreviousEventRow.N8RSI;
                                CurrentEventRow.N8NumTxPackets =
PreviousEventRow.N8NumTxPackets;
                                CurrentEventRow.N8NumTxFailures =
PreviousEventRow.N8NumTxFailures;
                                CurrentEventRow.N8NumRxPackets =
PreviousEventRow.N8NumRxPackets;
                                CurrentEventRow.NeighorHealthUpTime =
PreviousEventRow.NeighorHealthUpTime;
                    #endregion
                    }
                    #endregion

                    // Add EventRow to the EventRowList
                    EventRowList.Add(CurrentEventRow);
                    // Set CurrentEventRow to PreviousEventRow
                    PreviousEventRow = CurrentEventRow;
                    // Clear Current Event
                    CurrentEventRow = new EventRow();
                }
                // Done processing each event, Clear Current & Previous EventRows before
moving on to next Device
                CurrentEventRow = new EventRow();
                PreviousEventRow = new EventRow();
        }
        //Display the EventRowList Object
```

```
        EventRowList.GroupBy (erl => erl.MacAddress).Dump();
        //EventRowList.Dump();

        //Create csv file
        #region CSV File Creation
        FileStream fs = new FileStream("C:\\csv_datafinal.csv",
FileMode.OpenOrCreate);
         //var writer = new MemoryStream(;
        //populates csv file with row for each event
        foreach (var e in EventRowList)
        {
                  StringBuilder sb = new StringBuilder();
                            sb.Append(e.MacAddress.ToString());
                  sb.Append(String.Concat(",", e.DeviceType.ToString()));
                  sb.Append(String.Concat(",",e.EventTime.ToString()));
                  sb.Append(String.Concat(",", e.EventName.ToString()));
                  sb.Append(String.Concat(",", e.Latitude.ToString()));
                  sb.Append(String.Concat(",", e.Longitude.ToString()));
                  sb.Append(String.Concat(",", e.LocationUpTime.ToString()));
                  if (e.PlaceMark1 != null)
                  {
                      sb.Append(String.Concat(",", e.PlaceMark1));
                  }
                  if (e.PlaceMark2 != null)
                  {
                      sb.Append(String.Concat(",", e.PlaceMark2.ToString()));
                  }
                  sb.Append(String.Concat(",", e.IsMoving.ToString()));
                  if (e.PathSource != null)
                  {
                      sb.Append(String.Concat(",", e.PathSource.ToString()));
                  }
                  if (e.PathDestination != null)
                  {
                      sb.Append(String.Concat(",", e.PathDestination.ToString()));
                  }
                  if (e.Direction != null)
                  {
                      sb.Append(String.Concat(",", e.Direction.ToString()));
                  }

                            sb.Append(String.Concat(",", e.Charge.ToString()));
                            sb.Append(String.Concat(",", e.Temperature.ToString()));
                            sb.Append(String.Concat(",", e.Voltage.ToString()));
                            sb.Append(String.Concat(",", e.NumTxOk.ToString()));
                            sb.Append(String.Concat(",", e.NumTxFailure.ToString()));
                            sb.Append(String.Concat(",", e.NumRxOk.ToString()));
                            sb.Append(String.Concat(",", e.NumRxFailure.ToString()));
                            if (e.N1MacAddress != null)
                            {
                                  sb.Append(String.Concat(",",
e.N1MacAddress.ToString()));
                                  sb.Append(String.Concat(",", e.N1RSI.ToString()));
                                  sb.Append(String.Concat(",",
e.N1NumTxPackets.ToString()));
```

```
                                sb.Append(String.Concat(",",
e.N1NumRxPackets.ToString()));
                            }
                            if (e.N2MacAddress != null)
                            {
                                sb.Append(String.Concat(",",
e.N2MacAddress.ToString()));
                                sb.Append(String.Concat(",", e.N2RSI.ToString()));
                                sb.Append(String.Concat(",",
e.N2NumTxPackets.ToString()));
                                sb.Append(String.Concat(",",
e.N2NumTxFailures.ToString()));
                                sb.Append(String.Concat(",",
e.N2NumRxPackets.ToString()));
                            }
                            if (e.N3MacAddress != null)
                            {
                                sb.Append(String.Concat(",",
e.N3MacAddress.ToString()));
                                sb.Append(String.Concat(",", e.N3RSI.ToString()));
                                sb.Append(String.Concat(",",
e.N3NumTxPackets.ToString()));
                                sb.Append(String.Concat(",",
e.N3NumTxFailures.ToString()));
                                sb.Append(String.Concat(",",
e.N3NumRxPackets.ToString()));
                            }
                            if (e.N4MacAddress != null)
                            {
                                sb.Append(String.Concat(",",
e.N4MacAddress.ToString()));
                                sb.Append(String.Concat(",", e.N4RSI.ToString()));
                                sb.Append(String.Concat(",",
e.N4NumTxPackets.ToString()));
                                sb.Append(String.Concat(",",
e.N4NumTxFailures.ToString()));
                                sb.Append(String.Concat(",",
e.N4NumRxPackets.ToString()));
                            }
                            if (e.N5MacAddress != null)
                            {
                                sb.Append(String.Concat(",",
e.N5MacAddress.ToString()));
                                sb.Append(String.Concat(",", e.N5RSI.ToString()));
                                sb.Append(String.Concat(",",
e.N5NumTxPackets.ToString()));
                                sb.Append(String.Concat(",",
e.N5NumTxFailures.ToString()));
                                sb.Append(String.Concat(",",
e.N5NumRxPackets.ToString()));
                            }
                            if (e.N6MacAddress != null)
                            {
                                sb.Append(String.Concat(",",
e.N6MacAddress.ToString()));
```

```
                                    sb.Append(String.Concat(",", e.N6RSI.ToString()));
                                    sb.Append(String.Concat(",",
e.N6NumTxPackets.ToString()));
                                    sb.Append(String.Concat(",",
e.N6NumTxFailures.ToString()));
                                    sb.Append(String.Concat(",",
e.N6NumRxPackets.ToString()));
                              }
                              if (e.N7MacAddress != null)
                              {
                                    sb.Append(String.Concat(",",
e.N7MacAddress.ToString()));
                                    sb.Append(String.Concat(",", e.N7RSI.ToString()));
                                    sb.Append(String.Concat(",",
e.N7NumTxPackets.ToString()));
                                    sb.Append(String.Concat(",",
e.N7NumTxFailures.ToString()));
                                    sb.Append(String.Concat(",",
e.N7NumRxPackets.ToString()));
                              }
                              if (e.N8MacAddress != null)
                              {
                                    sb.Append(String.Concat(",",
e.N8MacAddress.ToString()));
                                    sb.Append(String.Concat(",", e.N8RSI.ToString()));
                                    sb.Append(String.Concat(",",
e.N8NumTxPackets.ToString()));
                                    sb.Append(String.Concat(",",
e.N8NumTxFailures.ToString()));
                                    sb.Append(String.Concat(",",
e.N8NumRxPackets.ToString()));
                              }
                              sb.Append(String.Concat(",",
e.NeighorHealthUpTime.ToString()));

                  sb.Append(Environment.NewLine);

                     var lineBytes = System.Text.Encoding.UTF8.GetBytes(sb.ToString());

                     fs.Write(lineBytes, 0, lineBytes.Length);
            }
            #endregion

}

public List<AssetMap> GetAssetMapping()
{
            List<AssetMap> AssetMapList = new List<AssetMap>();

            var AssetMappings = Events
                  .Where (e => e.Name.Contains("Node Discovery"))
                  .Where (e => e.EventDate > DateTime.Now.AddDays(-3))
                  .Select (e => new {Mac = e.Asset.Name, NodeID =
e.AdditionalProperties.First (ap => ap.Value != "Operational").Value})
                  .Distinct();
```

```
        foreach (var AM in AssetMappings)
        {
                AssetMap thisMap = new AssetMap() { Mac = AM.Mac, NodeID =
Int32.Parse(AM.NodeID)};
                AssetMapList.Add(thisMap);
        }

        return AssetMapList;
}

public List<Placemark> GetPlaceMarkList(String kmlFile)
{
        FileStream fs = new FileStream(kmlFile, FileMode.Open);
        KmlFile file = KmlFile.Load(fs);

        List<Placemark> Placemarks = new List<Placemark>();
        Kml kml = file.Root as Kml;
        if (kml != null)
        {
        foreach (var placemark in kml.Flatten().OfType<Placemark>())
        {
                Placemarks.Add(placemark);
        }
        }
        return Placemarks;
}

#region Device Classifications
public List<String> staticDevices = new List<String> {
"00-17-0D-00-00-38-26-2B",
"00-17-0D-00-00-38-21-68",
"00-17-0D-00-00-38-20-DB",
"00-17-0D-00-00-38-26-2E",
"00-17-0D-00-00-38-25-4E",
"00-17-0D-00-00-38-20-E8",
"00-17-0D-00-00-38-21-41",
"00-17-0D-00-00-38-21-61",
"00-17-0D-00-00-38-20-ED",
"00-17-0D-00-00-38-21-16",
"00-17-0D-00-00-38-20-F1"
};

public List<String> mobileDevices = new List<String> {
"00-17-0D-00-00-38-20-DA",
"00-17-0D-00-00-38-26-1A",
"00-17-0D-00-00-38-24-F8",
"00-17-0D-00-00-38-21-15",
"00-17-0D-00-00-38-21-0A",
"00-17-0D-00-00-38-25-54",
"00-17-0D-00-00-38-21-A5",
"00-17-0D-00-00-38-21-13",
"00-17-0D-00-00-38-21-2D",
"00-17-0D-00-00-38-21-7D",
"00-17-0D-00-00-38-21-63",
```

```
"00-17-0D-00-00-38-21-4D",
"00-17-0D-00-00-38-21-04",
"00-17-0D-00-00-38-26-08",
"00-17-0D-00-00-38-21-3E",
"00-17-0D-00-00-38-20-E7",
"00-17-0D-00-00-38-21-21",
"00-17-0D-00-00-38-20-F6",
"00-17-0D-00-00-38-21-43"
};
#endregion

public class EventRow
{
        // Event Info
        public string MacAddress;
        public string DeviceType;
        public DateTime EventTime;
        public string EventName;

        // Location Info
        public decimal? Latitude;
        public decimal? Longitude;
        public DateTime? LocationUpTime;
        public string PlaceMark1;
        public string PlaceMark2;
        public bool IsMoving;

        // Path Info
        public string PathSource;
        public string PathDestination;
        public string Direction;

        // Network Info
        //     public int NumMotes;
        //     public int AsnSize;
        //     public int NetReliability;
        //     public int NetPathStability;
        //     public int NetLatency;

        // Device Health Info
        public decimal? Charge;
        public decimal? Temperature;
        public decimal? Voltage;
        public decimal? NumTxOk;
        public decimal? NumTxFailure;
        public decimal? NumRxOk;
        public decimal? NumRxFailure;
        public DateTime? DeviceHealthUpTime;

        // Neighbor Health Info
        public string   N1MacAddress;
        public decimal? N1RSI;
        public decimal? N1NumTxPackets;
        public decimal? N1NumTxFailures;
        public decimal? N1NumRxPackets;
```

```
        public decimal? N1NumRxFailures;

        public string   N2MacAddress;
        public decimal? N2RSI;
        public decimal? N2NumTxPackets;
        public decimal? N2NumTxFailures;
        public decimal? N2NumRxPackets;
        public decimal? N2NumRxFailures;

        public string   N3MacAddress;
        public decimal? N3RSI;
        public decimal? N3NumTxPackets;
        public decimal? N3NumTxFailures;
        public decimal? N3NumRxPackets;
        public decimal? N3NumRxFailures;

        public string   N4MacAddress;
        public decimal? N4RSI;
        public decimal? N4NumTxPackets;
        public decimal? N4NumTxFailures;
        public decimal? N4NumRxPackets;
        public decimal? N4NumRxFailures;

        public string   N5MacAddress;
        public decimal? N5RSI;
        public decimal? N5NumTxPackets;
        public decimal? N5NumTxFailures;
        public decimal? N5NumRxPackets;
        public decimal? N5NumRxFailures;

        public string   N6MacAddress;
        public decimal? N6RSI;
        public decimal? N6NumTxPackets;
        public decimal? N6NumTxFailures;
        public decimal? N6NumRxPackets;
        public decimal? N6NumRxFailures;

        public string   N7MacAddress;
        public decimal? N7RSI;
        public decimal? N7NumTxPackets;
        public decimal? N7NumTxFailures;
        public decimal? N7NumRxPackets;
        public decimal? N7NumRxFailures;

        public string   N8MacAddress;
        public decimal? N8RSI;
        public decimal? N8NumTxPackets;
        public decimal? N8NumTxFailures;
        public decimal? N8NumRxPackets;
        public decimal? N8NumRxFailures;

        public DateTime? NeighorHealthUpTime;
}

public class NeighborStatus
```

```
{
	public int NeighborID;
	public int NeighborFlag;
	public int RRSI;
	public int NumTxPackets;
	public int NumTxFailures;
	public int NumRxPackets;
}

public class AssetMap
{
	public string Mac;
	public int NodeID;
}
```