Appendix

Sample Custom Default Query Processor

This section includes the complete sample code discussed in section *Create the Query Processor DLL*. The following class represents the main class used for custom Query Processing. It contains the processQueryResults method discussed in section *Create the Data Processor Method* and the build method discussed in section *Creating a list of Product Occurrences.*

```
using System;

using System.Collections.Generic;

using Aras.DynamicModelViewer.DataModel; // Required for Product Occurrences

using Aras.Server.Core.QueryBuilder; // Required to Process Query results

using System.Drawing; // Required for Colors used in Rendering Configurations

namespace CustomDefaultQP

{

 ///

 /// Main Class for processing the results from the execution of the default

 /// Query Definition used for Dynamic Visualization

 ///

 public class DefaultQueryResultsParser

 {

 ///

 /// Default Constructor

 ///

 public DefaultQueryResultsParser() { }

 ///

 /// Processes the given Query Result Items to build a hierarch of CAD Items

 ///
```


```
/// Input set of top-level Query Results
/// List of Product Occurrences
public ICollection processQueryResults(IEnumerable resultItems)
{
// Process each root CAD Item in the query results
List rootCADItems = new List();
foreach (var resultItem in resultItems)
{
if (resultItem.QueryItem.Alias == "CAD")
{
CAD nextCAD = new CAD();
nextCAD.processCAD(resultItem); // recurses through full hierarchy
rootCADItems.Add(nextCAD);
} // if()
} // foreach()
// Build the Product Occurrences for each root CAD Item
ICollection poList = new List();
foreach (CAD c in rootCADItems)
{
ProductOccurrenceInstance poInst = new ProductOccurrenceInstance();
poInst.Attributes.Add(new ProductOccurrenceAttr("QUERY ITEM REF ID", c.QryRefId));
poInst.Attributes.Add(new ProductOccurrenceAttr("ITEM ID", c.ID));
poInst.Name = c.Name;
if (c.Children.Count == 0 && c.SCFileID != null) // Assume this is a component part
{
poInst.ProductOccurrenceSource.Name = c.SCFileID; // use File Item Id for name
poInst.ProductOccurrenceSource.FileReferenceByTypeMap.Add(SourceModelType.Scs, c.SCFileID);
```

```
}
if (c.Children.Count == 0 && c.SCFileID == null)
poInst = null; // Invalid - no view file attached
else
{
poInst.SetAsRoot(); // root assembly,
poInst.SetServiceProperty(c.QryRefId, c.ID); // add service properties
poList.Add(poInst); // add to the PO list
}
// Recurse through child hierarchy
foreach (CAD child in c.Children)
build(poInst, child);
} // foreach (CAD c in rootCADItems)
return poList;
} // processQueryResults()
///
/// Constructs a list of Product Occurrences (PO) from the CAD Items parsed
/// from the Query Results. This will construct a unique Product Occurrence
/// for each instance of a part.
///
///  Product Occurrence to attach newly created POs to.
/// This value can't be null
/// CAD Item representing the child CAD data to buld from
private void build(ProductOccurrenceInstance parent, CAD c)
{
// Create a Product Occurrence for each instance of the given CAD
foreach (CADInstanceInfo cadInstInfo in c.Instances)
```

```
 {
 ProductOccurrenceInstance poInst = new ProductOccurrenceInstance();
 poInst.Attributes.Add(new ProductOccurrenceAttr("QUERY ITEM REF ID", c.QryRefId));
 poInst.Attributes.Add(new ProductOccurrenceAttr("ITEM ID", c.ID));
 // bounding box
 poInst.ProductOccurrenceSource.BoundingBox.Max.X = c.BBox.MaxX;
 poInst.ProductOccurrenceSource.BoundingBox.Max.Y = c.BBox.MaxY;
 poInst.ProductOccurrenceSource.BoundingBox.Max.Z = c.BBox.MaxZ;
 poInst.ProductOccurrenceSource.BoundingBox.Min.X = c.BBox.MinX;
 poInst.ProductOccurrenceSource.BoundingBox.Min.Y = c.BBox.MinY;
 poInst.ProductOccurrenceSource.BoundingBox.Min.Z = c.BBox.MinZ;
 // *** Testing Rendering Configuration ***
 ProductOccurrenceRenderingConfiguration rConfig = new ProductOccurrenceRenderingConfiguration();
 rConfig.SetColor(Color.FromArgb(55, 40, 0));
 rConfig.Opacity = 0.4;
 rConfig.Name = "test";
 poInst.RenderingConfigurations.Add(rConfig);
 // *** Testing Rendering Configuration ***
// **** ADD SetServiceProperty METHOD CALLS ****
 poInst.TransformationMatrix = cadInstInfo.XForm;
 poInst.SetServiceProperty(cadInstInfo.RefID, cadInstInfo.ID);
 poInst.SetServiceProperty(c.CADStrQryRefId, c.CADStructureID);
 poInst.SetServiceProperty(c.QryRefId, c.ID);
 // **** ADD SetServiceProperty METHOD CALLS ****
 poInst.Name = c.Name;
 if (c.Children.Count == 0 && c.SCFileID != null) // Assume this is a component part
 {
```

```
 poInst.ProductOccurrenceSource.Name = c.SCFileID; // use File Item Id for name
 poInst.ProductOccurrenceSource.FileReferenceByTypeMap.Add(SourceModelType.Scs, c.SCFileID);
 }
 if (c.Children.Count == 0 && c.SCFileID == null)
 poInst = null; // Invalid - no view file attached
 else
 parent.Children.Add(poInst);
 // Recurse through child hierarchy
 foreach (CAD child in c.Children)
 build(poInst, child);
 } // foreach(String xFormStr in c.Instances)
 } // build()
 } // class DefaultQueryResultsParser
} // namespace CustomDefaultQP
```

The following CAD class is used to collect information from the Query Results. It contains the processCAD method discussed in section *Processing Query Results*.

```
using System;
using System.Collections.Generic;
using Aras.Server.Core.QueryBuilder;
namespace CustomDefaultQP
{
 struct BoundingBox
 {
 public double MinX; // X Coord for minimum point
 public double MaxX; // X Coord for maximum point
 public double MinY; // Y Coord for minimum point
 public double MaxY; // Y Coord for maximum point
```

```
public double MinZ; // Z Coord for minimum point

public double MaxZ; // Z Coord for maximum point

} // class BoundingBox

///

/// Stores information for each CAD Instance Query Item

///

.

class CADInstanceInfo

{

public CADInstanceInfo()

{

XForm = "1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1"; // Default Identity Matrix

ID = "0";

}

///

/// CAD Instance transformation matrix

///

internal String XForm { get; set; }

///

/// CAD Instance Id

///

internal String ID { get; set; }

///

/// CAD Query Reference Id

///

internal String RefID { get; set; }

} // class CADInstanceInfo
```

```
///
/// Stores all data necessary for creating 3D View data assuming
/// it is contained within 'CAD' Query Items
///
class CAD
{
private BoundingBox _BBox; // Bounding volume
///
/// Default Constructor
///
internal CAD()
{
// Ensure that at least one instance is added
// This will be replaced in _processCADStructure() if instances
// are included in the query results
Instances.Add(new CADInstanceInfo());
} // CAD()
///
/// Processes a given CAD Item contained within the given QueryBuilderNode
///
/// Item containing properties for CAD Items
/// Required Properties not included in results
internal void processCAD(QueryBuilderNode qbItem)
{
QryRefId = qbItem.QueryItem.RefId;
try
{
```

```
ID = qbItem.GetProperty("id"); // required

Name = qbItem.GetProperty("name"); // required

}

catch

{

throw new ArgumentException("Query Definition is not defined properly: 'id' and 'name' Properties are requ

}

// Bounding Box Data. Alternate values ensure that there is a non-empty

// bounding volume defined

// NOTE: Bounding box data should first be retrieved from the

// related CAD ConversionInfo Relationship. If not present,

// retrieve from the CAD Item directly

_BBox.MinX = _getPropertyAsDouble(qbItem, "x_min", -1.0);

_BBox.MaxX = _getPropertyAsDouble(qbItem, "x_max", 1.0);

_BBox.MinY = _getPropertyAsDouble(qbItem, "y_min", -1.0);

_BBox.MaxY = _getPropertyAsDouble(qbItem, "y_max", 1.0);

_BBox.MinZ = _getPropertyAsDouble(qbItem, "z_min", -1.0);

_BBox.MaxZ = _getPropertyAsDouble(qbItem, "z_max", 1.0);

// for processing CAD children and associated view file

foreach (var child in qbItem.ChildNodes)

{

if (child.QueryItem.Alias == "CAD ConversionInfo")

_processCADConversionInfo(child);

if (child.QueryItem.Alias == "CAD Structure")

_processCADStructure(child);

if (child.QueryItem.Alias == "File")

_extractFileInfo(child);

} // foreach
```


```
} // processCAD(QueryBuilderNode qbItem)

///

/// Processes the bounding box data in the given CAD ConversionInfo node

///

///

private void _processCADConversionInfo(QueryBuilderNode qbItem)

{

_BBox.MinX = _getPropertyAsDouble(qbItem, "x_min", -1.0);

_BBox.MaxX = _getPropertyAsDouble(qbItem, "x_max", 1.0);

_BBox.MinY = _getPropertyAsDouble(qbItem, "y_min", -1.0);

_BBox.MaxY = _getPropertyAsDouble(qbItem, "y_max", 1.0);

_BBox.MinZ = _getPropertyAsDouble(qbItem, "z_min", -1.0);

_BBox.MaxZ = _getPropertyAsDouble(qbItem, "z_max", 1.0);

} // _processCADConversionInfo(QueryBuilderNode qbItem)

///

/// Returns the value of the given Property as a double, use given alternate

/// if the value is not set or is not defined.

///

/// Query Builder Item containing the Property

/// Property Name

/// Used when given property not set

/// Value if Property exists and is set, 'alt' value otherwise

private double _getPropertyAsDouble(QueryBuilderNode qbItem, String propName, double alt)

{

// check for existence of Property (in future)

if (qbItem.TryGetProperty(propName, out string tmpPropVal) && tmpPropVal != null)

return Double.Parse(tmpPropVal);

else
```


```
return alt;
} // getPropertyAsDouble()
///
/// Processes the child CAD Instances and Files associated with the given
/// QueryBuilderItem
///
///
/// Required Properties not included in results
private void _processCADStructure(QueryBuilderNode qbItem)
{
// Save the instances and store with child CAD Item
List curInstances = new List(); CAD nextCAD = new CAD();
nextCAD.CADStructureID = qbItem.ItemId;
nextCAD.CADStrQryRefId = qbItem.QueryItem.RefId;
// Loop through all CAD Instances and CAD Items
foreach (var child in qbItem.ChildNodes)
{
try
{
if (child.QueryItem.Alias == "CAD Instance" &&
child.TryGetProperty("transformation_matrix", out string tmpXForm) &&
tmpXForm != null) {
CADInstanceInfo cadInfo = new CADInstanceInfo();
cadInfo.XForm = tmpXForm;
cadInfo.ID = child.ItemId;
cadInfo.RefID = child.QueryItem.RefId;
curInstances.Add(cadInfo);
```

```
}
 if (child.QueryItem.Alias == "CAD")
 {
 nextCAD.processCAD(child);
 Children.Add(nextCAD);
 }
 } // try
 catch { }
 } // foreach()
 if (curInstances.Count > 0)
 nextCAD.Instances = curInstances;
 } // _processCADStructure(QueryBuilderNode qbItem)
 ///
 /// Returns the first found QueryBuilderNode with the
 /// given alias that is a descendant of the given Item. Note that if
 /// the given Item has the alias, it is returned
 ///
 /// Item containing descendants to search
 /// Alias Name of Item to search for
 /// NULL, if not found; first found node with given alias otherwise
 private QueryBuilderNode _findChildWithAlias(QueryBuilderNode qbItem, String alias)
 {
 if (qbItem.QueryItem.Alias == alias)
 return (qbItem);
 else
 {
 foreach (var child in qbItem.ChildNodes)
```

```
{
QueryBuilderNode descItem = _findChildWithAlias(child, alias);
if (descItem != null)
return (descItem);
} // for()
} // else
return (null);
} // findChildWithAlias(QueryBuilderNode qbItem, String alias)
///
/// Extracts the SCS file data from the given QueryBuilderNode. This
/// method assumes that the Properties for 'id' and 'filename' exist
/// in the given results node with alias 'File_1'
///
///
/// Required File Properties not included in results
private void _extractFileInfo(QueryBuilderNode qbItem)
{
QueryBuilderNode child = _findChildWithAlias(qbItem, "File_1");
if (child != null)
{
// filename and id are required for Product Occurrences
if (child.TryGetProperty("filename", out string tmpFileName) &&
tmpFileName != null &&
child.TryGetProperty("id", out string tmpId) &&
tmpId != null)
{
SCFile = tmpFileName;
```

```
SCFileID = tmpId;
}
else
throw new ArgumentException("Query Definition is not defined properly: 'filename' and 'id' Properties are
} // if (child != null)
} // _extractFileInfo(QueryBuilderItem qbItem)
///
/// Returns the name of the CAD Query Response Item
///
internal String Name { get; private set; }
///
/// Returns the Query Reference ID from the CAD Query Response Item
/// used to create this CAD Object
///
internal String QryRefId { get; private set; }
///
/// Returns the Query Reference ID from the CAD Structure Query
/// Response Item used to create this CAD Object
///
internal String CADStrQryRefId { get; private set; }
///
/// Returns the Id of the CAD Query Response Item
///
internal String ID { get; private set; }
///
/// Returns the Id of the CAD Query Response Item
///
```

```
 internal String CADStructureID { get; private set; }
 ///
 /// Returns the SCS/SCZ File name associated with the CAD Query Response Item
 ///
 internal String SCFile { get; private set; }
 ///
 /// Returns the SCS/SCZ File ID associated with the CAD Query Response Item
 ///
 internal String SCFileID { get; private set; }
 ///
 /// Returns the list of data for CAD Instance Query Response Items
 ///
 internal List Instances { get; private set;} = new List();
 ///
 /// List of child CAD Items as determined by the CAD Structure
 ///
 internal List Children { get; } = new List();
 ///
 /// Bounding Volume
 ///
 internal BoundingBox BBox { get { return _BBox; } }
 } // class CAD
} // namespace CustomDefaultQP
```