

The Steps and Changes for Bricscad V25 and V26.

MIT License

Copyright (c) 2026 Sakko

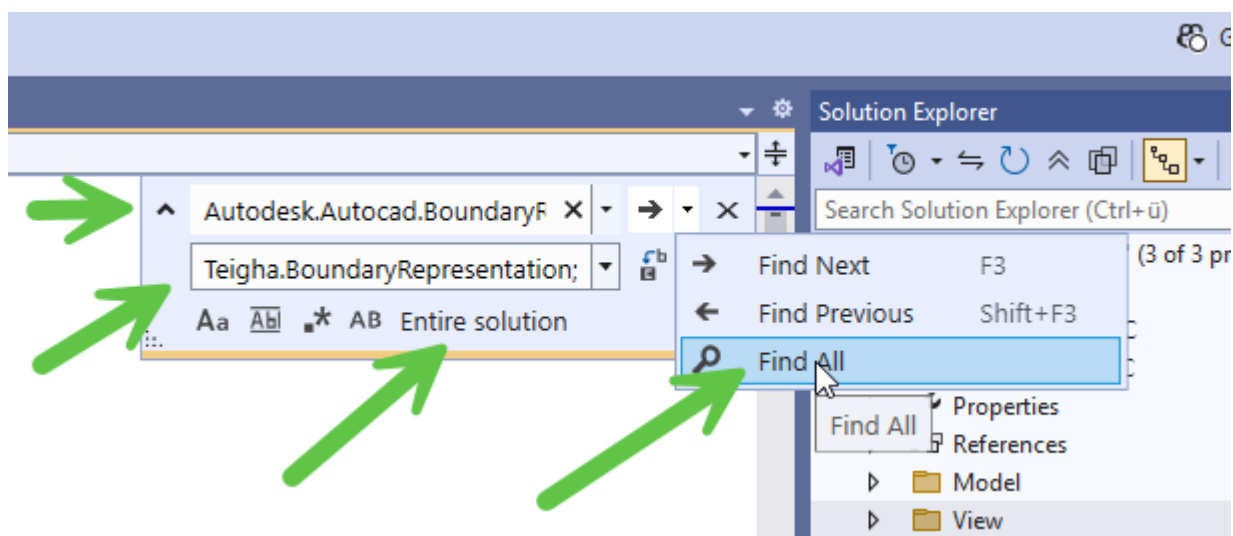
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

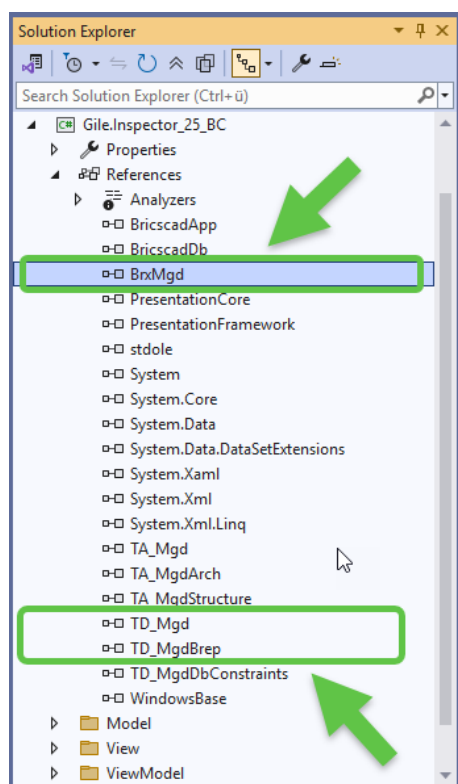
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.

Change all References to the managed .dll's in the code **and** in project explorer from AutoCAD to Bricscad. Just use the replace function in Visual Studio for the whole Solution....





```

1  using Teigha.BoundaryRepresentation;
2  using Teigha.DatabaseServices;
3  using Teigha.Geometry;
4  using Teigha.GraphicsInterface;
5  using Teigha.LayerManager;
6  using Teigha.Runtime;
7
8  using System;
9  using System.Collections;
10 using System.Collections.Generic;
11 using System.Collections.ObjectModel;
12 using System.ComponentModel;
13 using System.Data.Common;
14 using System.Linq;
15 using System.Reflection;
16
17 using AcAp = Bricscad.ApplicationServices.Application;
18 using AcDb = Teigha.DatabaseServices;
19

```

2.

In the file **InspectorViewModel.cs** the following lines should be excluded:

```
//case EdgeRef[] edges: items = fromIEnumerable(edges); break;
```

```

77
78
79  case BulgeVertexCollection collection: items = fromICollection<BulgeVer
80  case IEnumerable<DBObject> sequence: items = fromIEnumerable(sequence);
81  //case EdgeRef[] edges: items = fromIEnumerable(edges); break;
    case RowsCollection sequence: items = fromIEnumerable(sequence); break;
    case ColumnsCollection sequence: items = fromIEnumerable(sequence): bre

```

```
//case IEnumerable<Profile3d> collection: items = fromIEnumerable(collection); break;
```

```

164     else
165     {
166         switch (value)
167         {
168             case IEnumerable<Entity> collection: items = fromIEnumerable(collection); break;
169             //case IEnumerable<Profile3d> collection: items = fromIEnumerable(collection); break;
170             case Dictionary<string, string>.Enumerator dict: items = fromObject(dict); break;
171             case object[] objs: items = fromObject(objs); break;
172             default: break;
173         }
174     }

```

```

/*
    case BlockReference br when AssocArray.IsAssociativeArray(br.ObjectId):
        var parameters = AssocArray.GetAssociativeArray(br.ObjectId).GetParameters();
        yield return new PropertyItem("Associative array parameters", parameters,
typeof(BlockReference), true);
        break;
*/

```

```

406         break;
407         /*
408         case BlockReference br when AssocArray.IsAssociativeArray(br.ObjectId):
409             var parameters = AssocArray.GetAssociativeArray(br.ObjectId).GetParameters();
410             yield return new PropertyItem("Associative array parameters", parameters, typeof(BlockReference), true);
411             break;
412         */
413         default:
414             break;
415     }
416 }

```

3.

In the file **InspectableItem.cs** the following lines should be replaced with **non Linq equivalent code** because they generate exceptions:

```

// Using Linq generates exceptions in Bricscad !!!
/*
if (dbObj is SymbolTable table)
{
    Children = table
        .Cast<ObjectId>()
        .Select(x => new InspectableItem(x));
}
else if (dbObj is DBDictionary dict)
{
    if (id == id.Database.NamedObjectsDictionaryId)
    {
        Name = "Named Objects Dictionary";
        IsExpanded = true;
    }
    Children = ((DBDictionary)dbObj)
        .Cast<DictionaryEntry>()
        .Select(e => new InspectableItem((ObjectId)e.Value, name: (string)e.Key));
}

```

```

    }
    */

```

Replacement code:

```
/*
```

no-LINQ version of the following logic, written using plain foreach loops.

The structure is kept identical to the original code, just without Cast<> or Select.

Notes

SymbolTable implements IEnumerable, so you can iterate directly with foreach (ObjectId id in table).

DBDictionary implements IDictionary, so iterating with foreach (DictionaryEntry entry in dict) is correct. The resulting Children collection is identical to what LINQ produced.

If wanted, It is also possible to:

- wrap this into a helper method

- make Children lazy-evaluated

- add sorting or filtering

- or convert this into a pattern-matching switch expression (still without LINQ)

```
*/
```

```
if (dbObj is SymbolTable table)
```

```
{
```

```
    var list = new List<InspectableItem>();
```

```
    foreach (ObjectId oid in table)
```

```
    {
```

```
        list.Add(new InspectableItem(oid));
```

```
    }
```

```
    Children = list;
```

```
}
```

```
else if (dbObj is DBDictionary dict)
```

```
{
```

```
    if (id == id.Database.NamedObjectsDictionaryId)
```

```
    {
```

```
        Name = "Named Objects Dictionary";
```

```
        IsExpanded = true;
```

```
    }
```

```
    var list = new List<InspectableItem>();
```

```
    foreach (DictionaryEntry entry in dict)
```

```
    {
```

```
        ObjectId childId = (ObjectId)entry.Value;
```

```
        string key = (string)entry.Key;
```

```
        list.Add(new InspectableItem(childId, name: key));
```

```
    }
```

```

        Children = list;
    }

```

4. All references in code like :

if (type.Namespace != null && type.Namespace.StartsWith("Autodesk.AutoCAD"))

```

60      ↓ | | | | if (type.Namespace != null && type.Namespace.StartsWith("Autodesk.AutoCAD"))

```

should be replaced with

if (type.Namespace.StartsWith("Teigha") || type.Namespace.StartsWith("Bricscad"))

```

61      ↓ | | | | if (type.Namespace.StartsWith("Teigha") || type.Namespace.StartsWith("Bricscad"))

```

5.

In the **ReferencedBy.cs** file enclose the following part of code in the If statement to avoid object specific exceptions (bim objects...?)

```

private void ProcessObject(Transaction tr, ObjectId curId)
{
    if (!curId.IsNull && !curId.IsErased)
    {
        var dbObj = tr.GetObject(curId, OpenMode.ForRead);
    }
}

```

```
var filer = new ReferenceFiler();
dbObj.DwgOut(filer);

    if (filer.HardOwnershipIds.Contains(sourceId))
        HardOwnershipIds.Add(dbObj.ObjectId);
    if (filer.SoftOwnershipIds.Contains(sourceId))
        SoftOwnershipIds.Add(dbObj.ObjectId);
    if (filer.HardPointerIds.Contains(sourceId))
        HardPointerIds.Add(dbObj.ObjectId);
    if (filer.SoftPointerIds.Contains(sourceId))
        SoftPointerIds.Add(dbObj.ObjectId);

    foreach (ObjectId id in filer.HardOwnershipIds)
    {
        ProcessObject(tr, id);
    }
    foreach (ObjectId id in filer.SoftOwnershipIds)
    {
        ProcessObject(tr, id);
    }
}
```