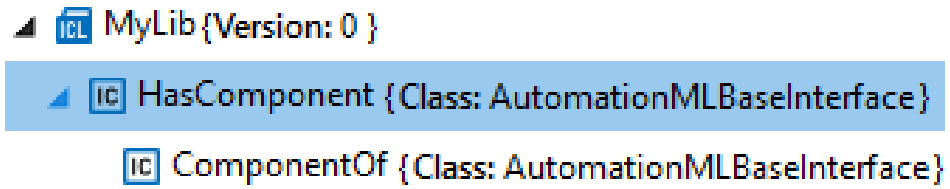
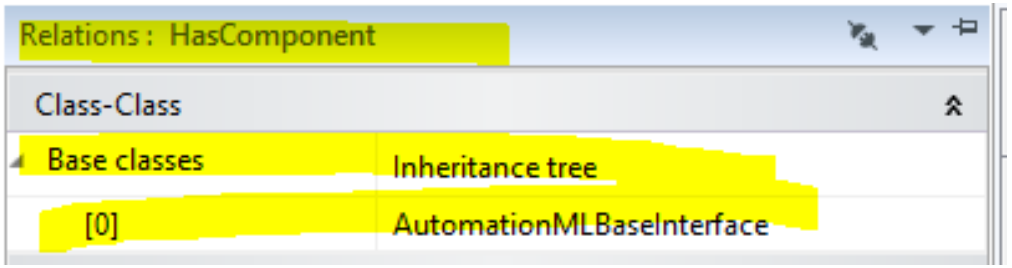
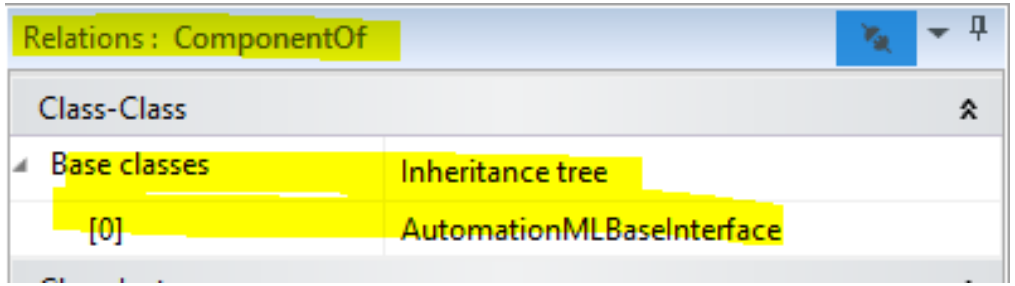
The AML file contains the following **InterfaceClassLib**:



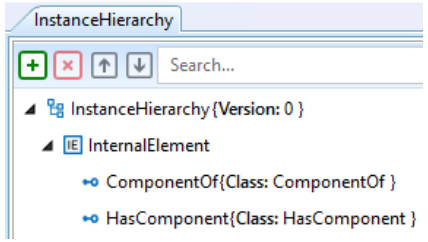
Important: Both interface classes are derived from **AutomationMLBaseInterface**.
This can be confirmed by examining the **inheritance tree** of each interface in the **AML Editor**.





ComponentOf is **not** derived from **HasComponent**.

The **AMLEngine** contains the function **IsDerivedFromInterfaceClass**, but it behaves incorrectly.
The following **InstanceHierarchy** is loaded using the AMLEngine:



When checking whether one of the interface classes (ComponentOf, HasComponent) is derived from HasComponent, the result incorrectly indicates that **both** interface classes are derived from HasComponent.

- **This is incorrect.**
- **The expected behavior** is that only HasComponent is returned, since ComponentOf is **not** derived from HasComponent.

What we need

A function that checks whether a given class is **the same type or a derived type** of another class.

Next steps

It should also be verified whether the same issue exists for other types, such as:

- RoleClass
- AttributeType
- SystemUnitClass

This behavior can be verified using the following project.

```
1  // See https://aka.ms/new-console-template for more information
2  using Aml.Engine.CAEX;
3  using Aml.Engine.CAEX.Extensions;
4
5  var document = CAEXDocument.LoadFromFile("AmlEngineIsDerivedFromIssue.aml");
6  var iH = document.CAEXFile.InstanceHierarchy.FirstOrDefault();
7  var iE = iH?.InternalElement.FirstOrDefault();
8
9  string hasComponentCaexPath = @"MyLib/HasComponent";
10
11 var myderivedList = iE.ExternalInterface.Where((ExternalInterfaceType e) => e.IsDerivedFromInterfaceClass(hasComponentCaexPath));
12
13 foreach (var derived in myderivedList)
14 {
15     Console.WriteLine($"{derived.Name} is derived from / or of type {hasComponentCaexPath}");
16 }
```