Partial types

Partial types allow classes, structs, and interfaces to be broken into multiple pieces stored in different source files for easier development and maintenance.

Type modifier *partial* is used when defining a type in multiple parts. To ensure compatibility with existing programs, this modifier is different than other modifiers: like get and set , it is not a keyword, and it must appear immediately before one of the keywords class, struct , or interface.

Each part of a partial type declaration must include a Type modifier *partial* and must be declared in the same namespace as the other parts.
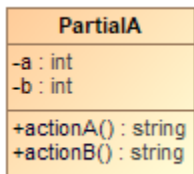
✓ All child elements will be merged into one Class element.

Example 1

```
//The partial class is written into one class file.
public partial class PartialA{
int a;
string actionA()
        {
        }
}

public partial class PartialA{
int b; string actionB()
        {
        }
}
```

Reversed UML model:

The partial modifier indicates that additional parts of the type declaration may exist elsewhere, but the existence of such additional parts is not a requirement. It is valid for just a single declaration of a type to include the partial modifier.

All parts of a partial type must be compiled together such that the parts can be merged at compile-time. Partial types specifically do not allow already compiled types to be extended.

✓ All child elements will be merged into one Class element, the same as both classes are written into one Class file.

Example 2

```
//The partial class is written into separate class file.
//PartialA1.cs
public partial class PartialA{
int a;
string actionA()
{
        }
}
//PartialA2.cs
public partial class PartialA{
        int b;
        string actionB()
        {
        }
}
```

Reversed UML model:

Nested types can be declared in multiple parts by using the Type modifier *partial*. Typically, the containing type is declared using partial as well, and each part of the nested type is declared in a different part of the containing type.

✓

All inner classes have to be located as inner class of the parent class (partial class).
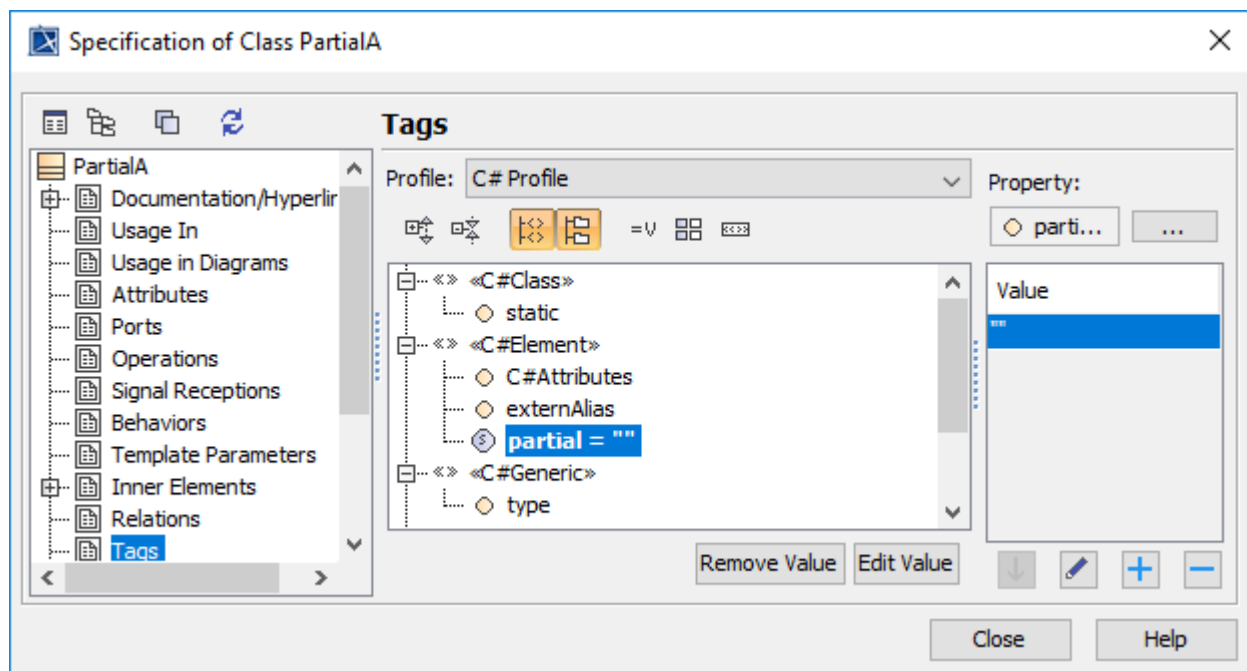
Example 3

```
//The partial class with inner class
public partial class PartialA{
        public class B
        {
                int b;
        }
}
public partial class PartialA{
        int a;
        string actionB()
        {
        }
        public class C
        {
                int c;
        }
}
```
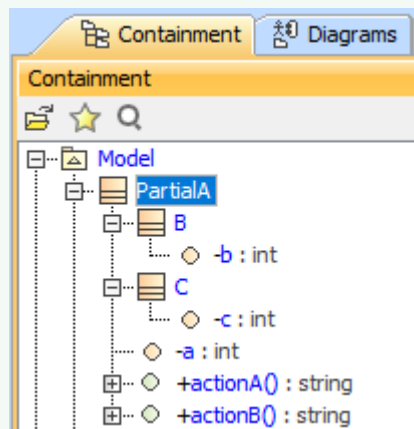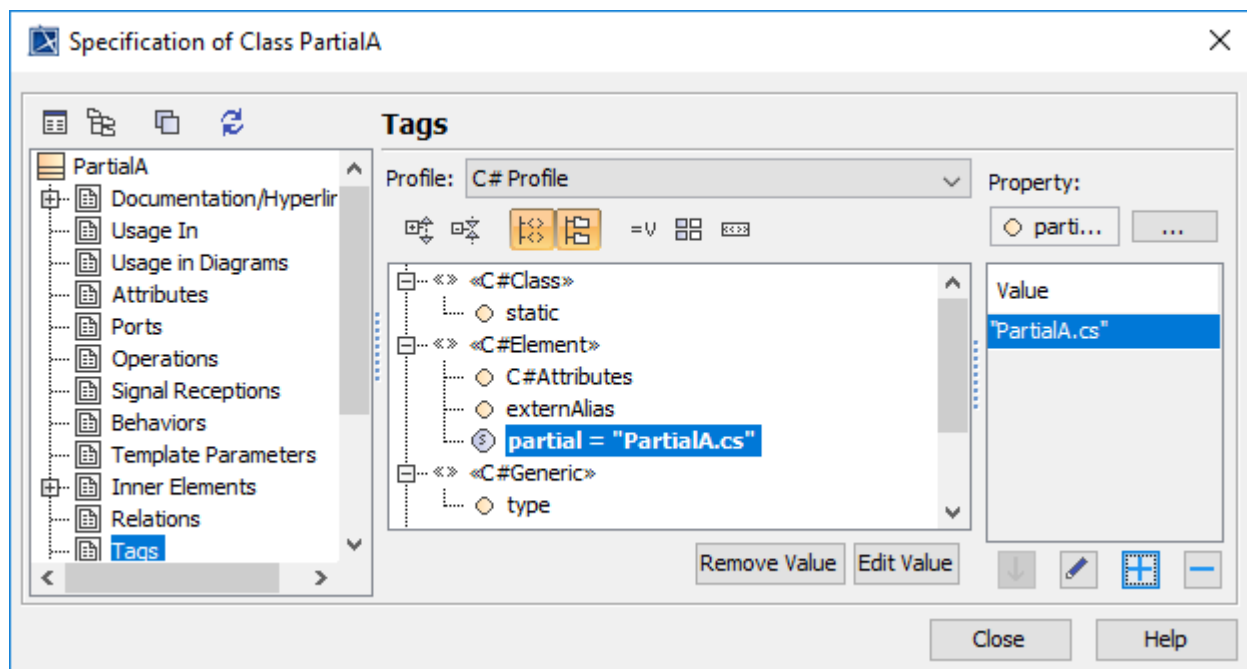

The type modifier *partial* is not permitted on delegate or enum declarations.

The Tag *partial* will have a blank tagged value for Partial Class Element:

Partial Class Element with blank **Tag** partial value

The Tag *partial* will have a value of the file name that the class belongs to for each child element in Partial Class Element:

Partial Class Element with **Tag** partial "PartialA.cs" value