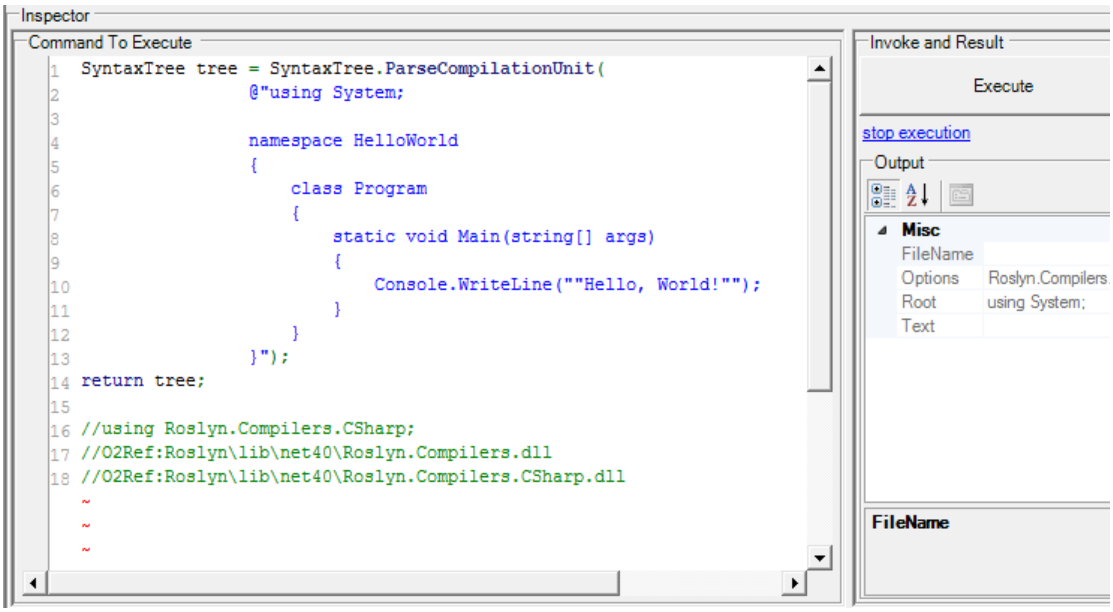
Using Roslyn (consuming Static and Sematic data)

Executing scripts inside document:

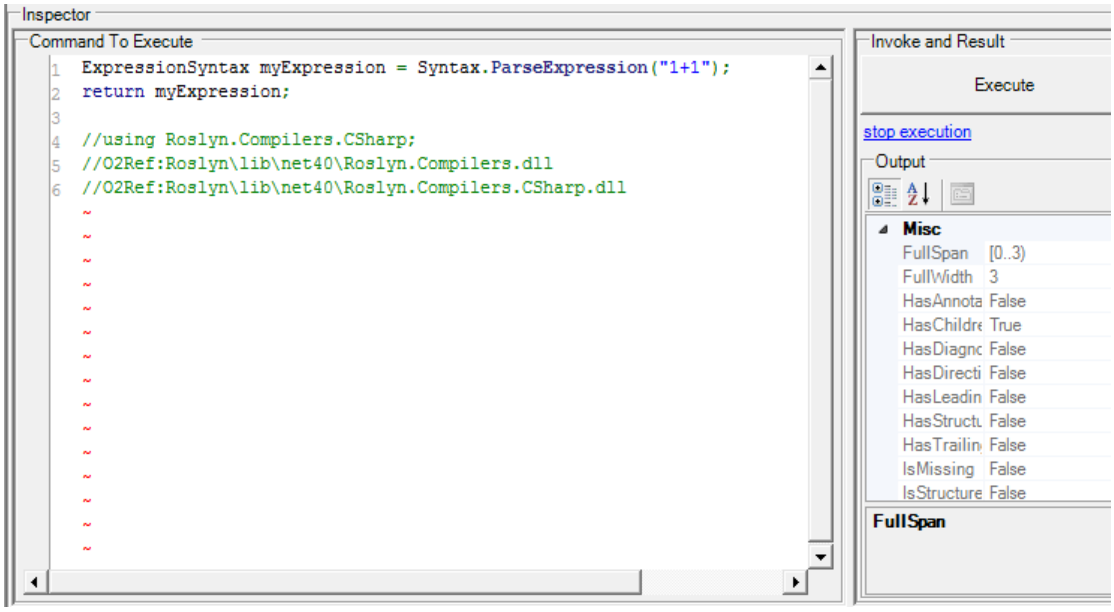
Exposing the C# and VB compiler's code analysis http://msdn.microsoft.com/en-us/hh500769#Toc306015665

3.2 Obtaining a Syntax Tree

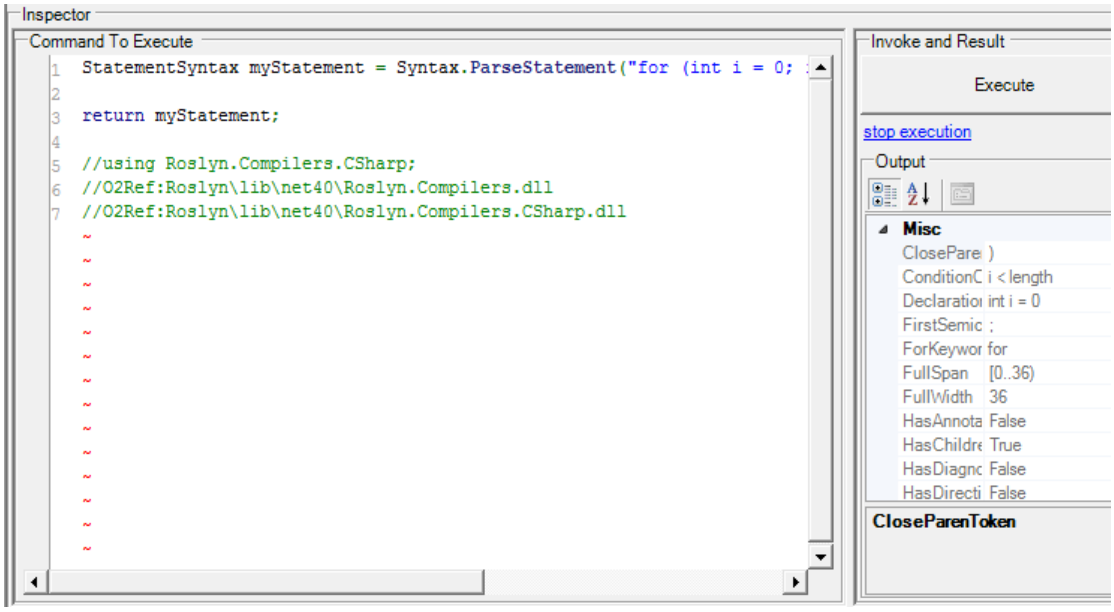
```
SyntaxTree tree = SyntaxTree.ParseCompilationUnit(
                @"using System;

                namespace HelloWorld
                {
                    class Program
                    {
                        static void Main(string[] args)
                        {
                            Console.WriteLine(""Hello, World!"");
                        }
                    }
                }");
return tree;

//using Roslyn.Compilers.CSharp;
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```




```
ExpressionSyntax myExpression = Syntax.ParseExpression("1+1");
return myExpression;

//using Roslyn.Compilers.CSharp;
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```



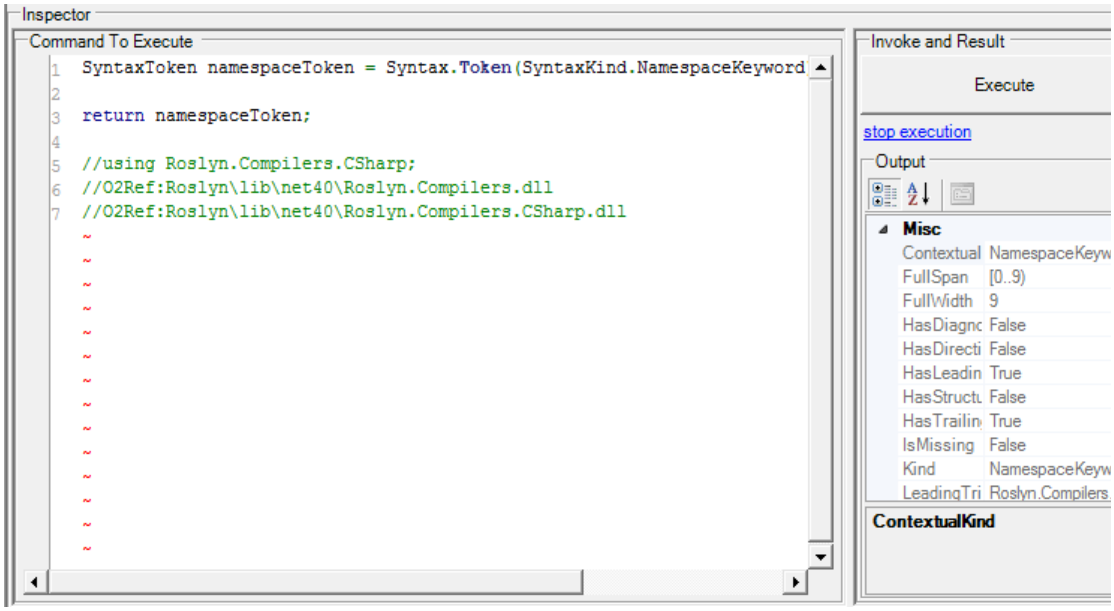

```
StatementSyntax myStatement = Syntax.ParseStatement("for (int i = 0; i <
length; i++) { }");

return myStatement;

//using Roslyn.Compilers.CSharp;
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```



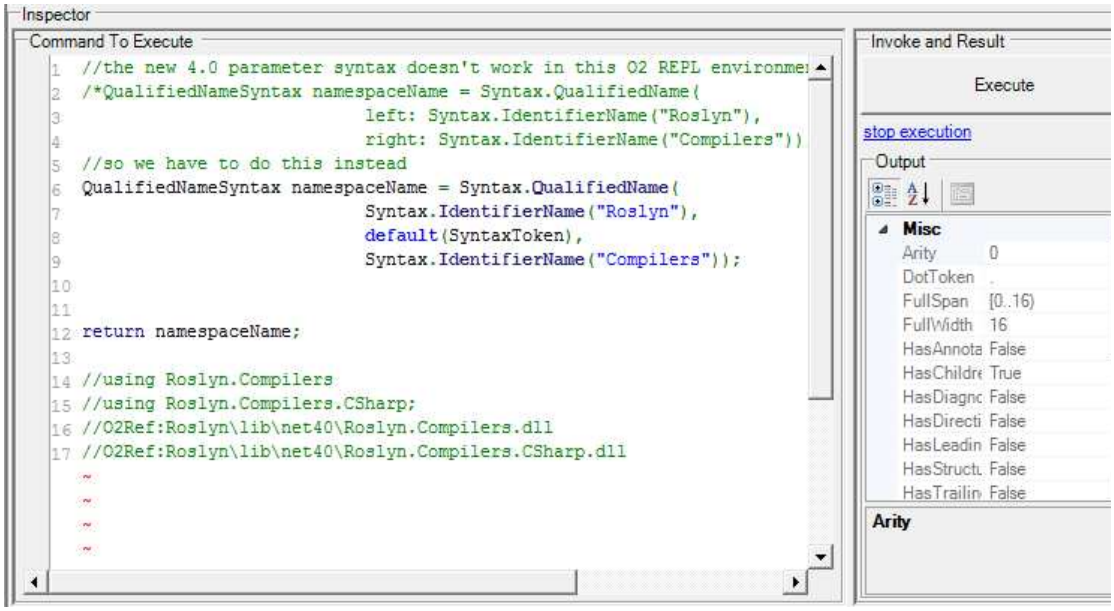
3.3 Manipulating Syntax Nodes

```
SyntaxToken namespaceToken = Syntax.Token(SyntaxKind.NamespaceKeyword);

return namespaceToken;

//using Roslyn.Compilers.CSharp;
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
```

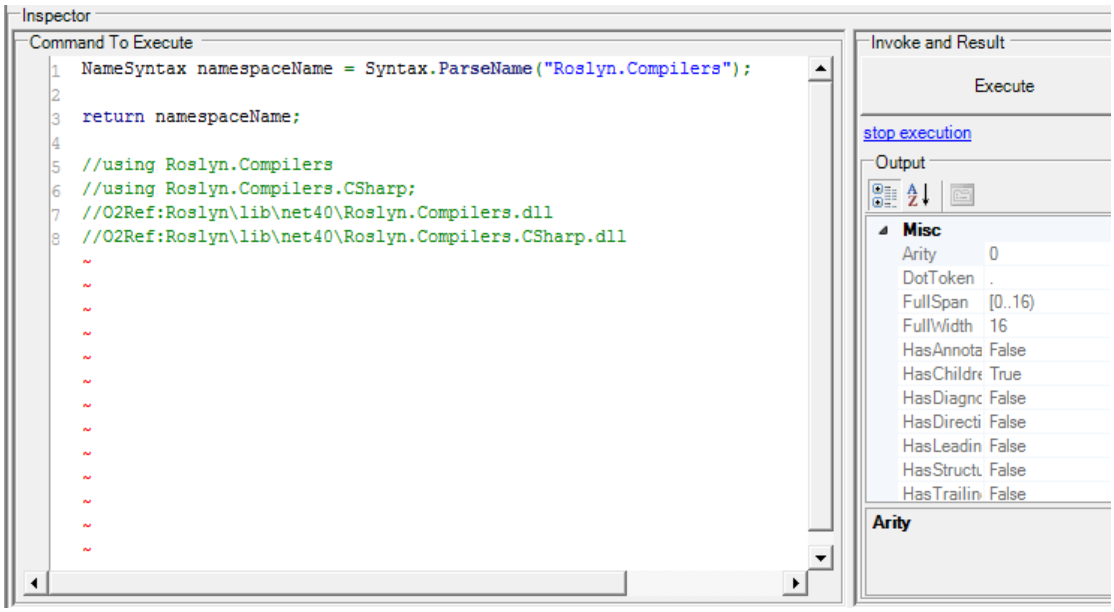
```
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```




```
//the new 4.0 parameter syntax doesn't work in this O2 REPL environment
/*QualifiedNameSyntax namespaceName = Syntax.QualifiedName(
                            left: Syntax.IdentifierName("Roslyn"),
                            right: Syntax.IdentifierName("Compilers"));*/
//so we have to do this instead
QualifiedNameSyntax namespaceName = Syntax.QualifiedName(
                            Syntax.IdentifierName("Roslyn"),
                            default(SyntaxToken),
                            Syntax.IdentifierName("Compilers"));


return namespaceName;

//using Roslyn.Compilers
 //using Roslyn.Compilers.CSharp;
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```

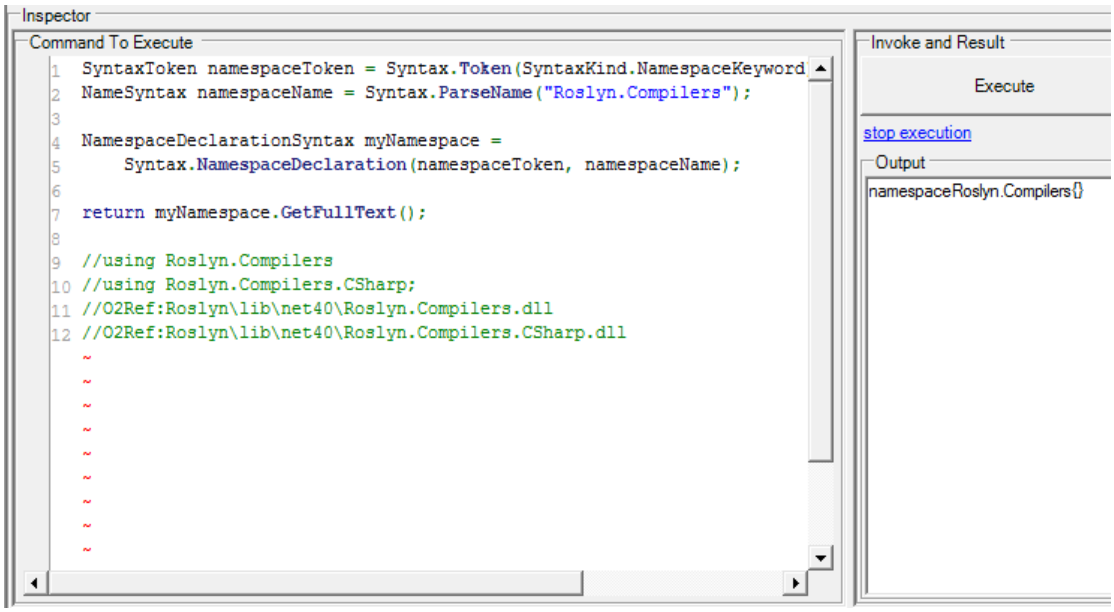



```
NameSyntax namespaceName = Syntax.ParseName("Roslyn.Compilers");

return namespaceName;

//using Roslyn.Compilers
 //using Roslyn.Compilers.CSharp;
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```

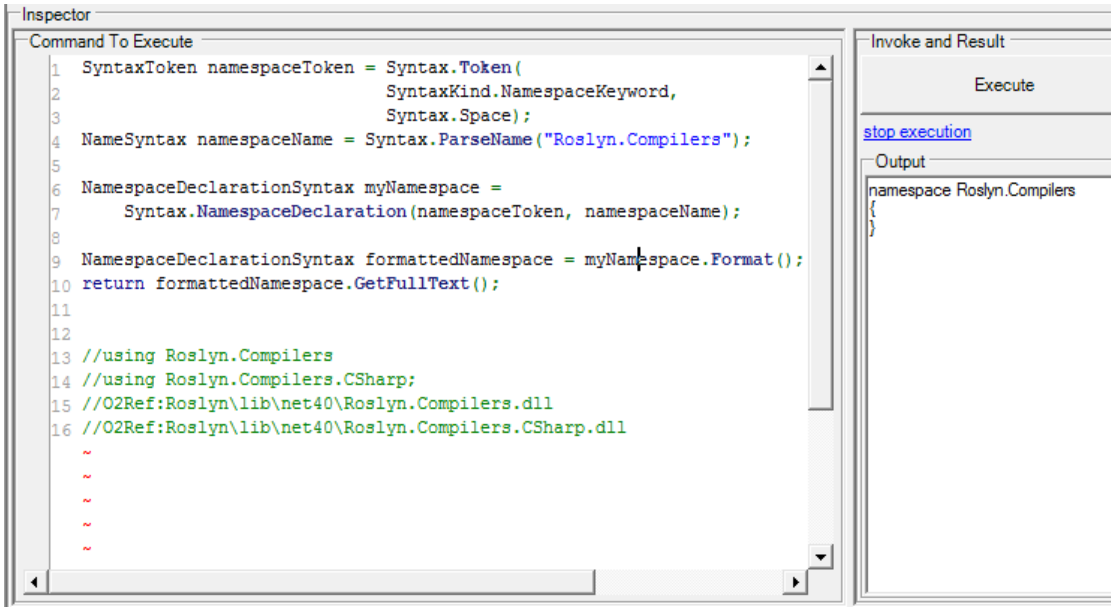



```
SyntaxToken namespaceToken = Syntax.Token(SyntaxKind.NamespaceKeyword);
NameSyntax namespaceName = Syntax.ParseName("Roslyn.Compilers");

NamespaceDeclarationSyntax myNamespace =
       Syntax.NamespaceDeclaration(namespaceToken, namespaceName);
```

```
return myNamespace.GetFullText();

//using Roslyn.Compilers
 //using Roslyn.Compilers.CSharp;
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```

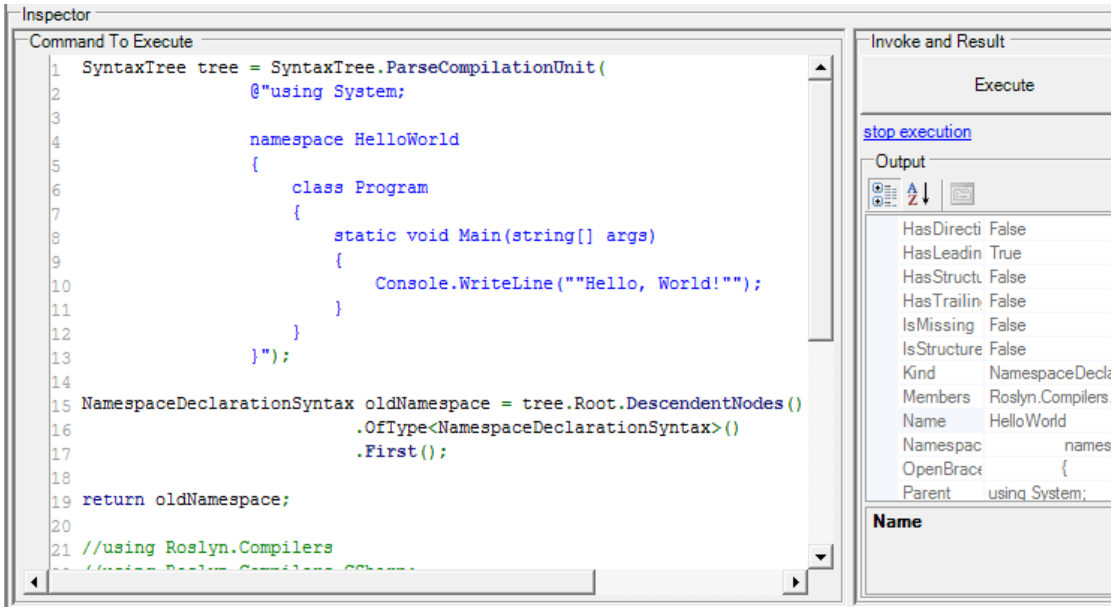



```
SyntaxToken namespaceToken = Syntax.Token(
                             SyntaxKind.NamespaceKeyword,
                             Syntax.Space);
NameSyntax namespaceName = Syntax.ParseName("Roslyn.Compilers");

NamespaceDeclarationSyntax myNamespace =
      Syntax.NamespaceDeclaration(namespaceToken, namespaceName);

NamespaceDeclarationSyntax formattedNamespace = myNamespace.Format();
return formattedNamespace.GetFullText();


//using Roslyn.Compilers
 //using Roslyn.Compilers.CSharp;
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```

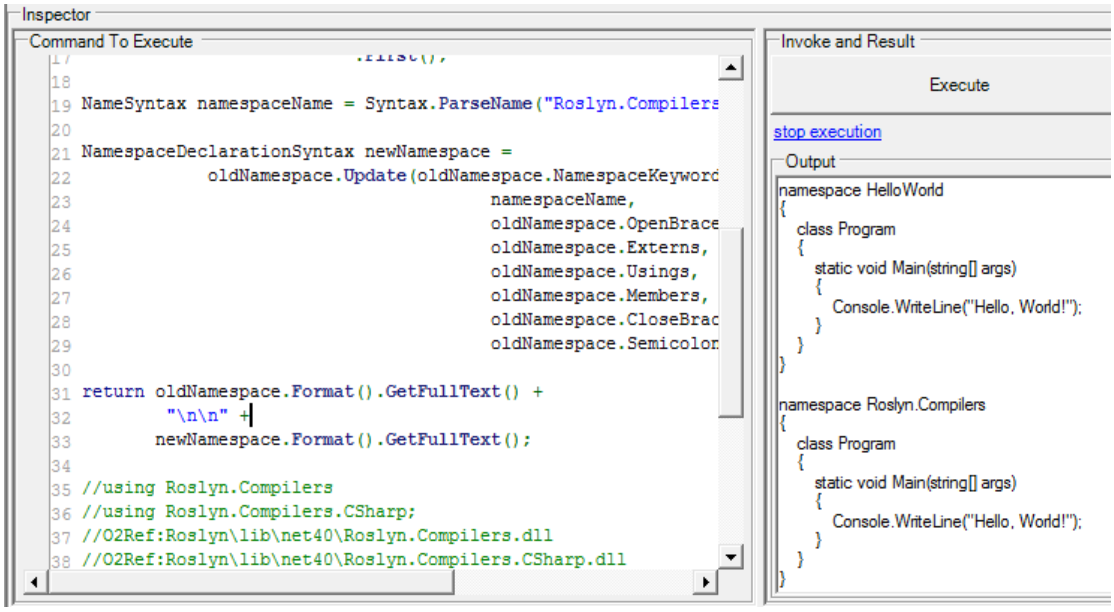


3.3.2 Modifying an existing node

```
SyntaxTree tree = SyntaxTree.ParseCompilationUnit(
                @"using System;

                namespace HelloWorld
                {
                    class Program
                    {
                        static void Main(string[] args)
                        {
                            Console.WriteLine(""Hello, World!"");
                        }
                    }
                }");

NamespaceDeclarationSyntax oldNamespace = tree.Root.DescendentNodes()
                          .OfType<NamespaceDeclarationSyntax>()
                          .First();

return oldNamespace;

//using Roslyn.Compilers
 //using Roslyn.Compilers.CSharp;
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```




```
SyntaxTree tree = SyntaxTree.ParseCompilationUnit(
                @"using System;

                namespace HelloWorld
                {
                    class Program
                    {
                        static void Main(string[] args)
                        {
                            Console.WriteLine(""Hello, World!"");
                        }
                    }
                }");

NamespaceDeclarationSyntax oldNamespace = tree.Root.DescendentNodes()
                          .OfType<NamespaceDeclarationSyntax>()
                          .First();

NameSyntax namespaceName = Syntax.ParseName("Roslyn.Compilers");

NamespaceDeclarationSyntax newNamespace =
                    oldNamespace.Update(oldNamespace.NamespaceKeyword,
                                        namespaceName,
                                        oldNamespace.OpenBraceToken,
                                        oldNamespace.Externs,
                                        oldNamespace.Usings,
                                        oldNamespace.Members,
                                        oldNamespace.CloseBraceToken,
                                        oldNamespace.SemicolonTokenOpt);

return oldNamespace.Format().GetFullText() +
         "\n\n" +
        newNamespace.Format().GetFullText();

//using Roslyn.Compilers
 //using Roslyn.Compilers.CSharp;
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```



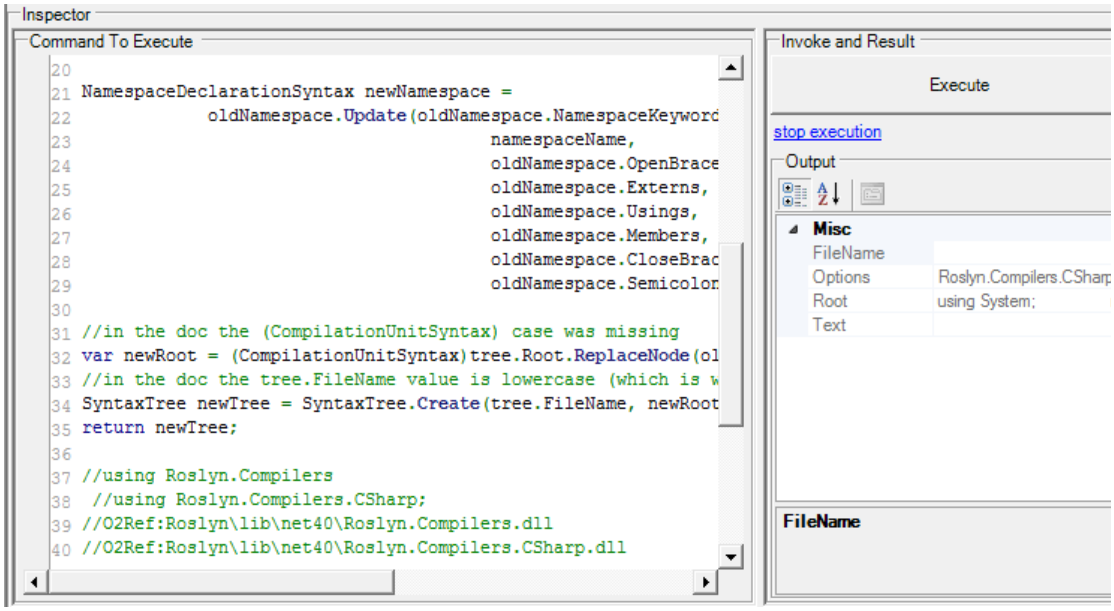
3.3.3 Replacing a node

```
SyntaxTree tree = SyntaxTree.ParseCompilationUnit(
                @"using System;

                namespace HelloWorld
                {
                    class Program
                    {
                        static void Main(string[] args)
                        {
                            Console.WriteLine(""Hello, World!"");
                        }
                    }
                }");

NamespaceDeclarationSyntax oldNamespace = tree.Root.DescendentNodes()
                          .OfType<NamespaceDeclarationSyntax>()
                          .First();

NameSyntax namespaceName = Syntax.ParseName("Roslyn.Compilers");

NamespaceDeclarationSyntax newNamespace =
                    oldNamespace.Update(oldNamespace.NamespaceKeyword,
                                        namespaceName,
                                        oldNamespace.OpenBraceToken,
                                        oldNamespace.Externs,
                                        oldNamespace.Usings,
                                        oldNamespace.Members,
                                        oldNamespace.CloseBraceToken,
                                        oldNamespace.SemicolonTokenOpt);

SyntaxNode newRoot = tree.Root.ReplaceNode(oldNamespace, newNamespace);

return newRoot.Format().GetFullText();

//using Roslyn.Compilers
 //using Roslyn.Compilers.CSharp;
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```

3.3.4 Creating a Syntax Tree

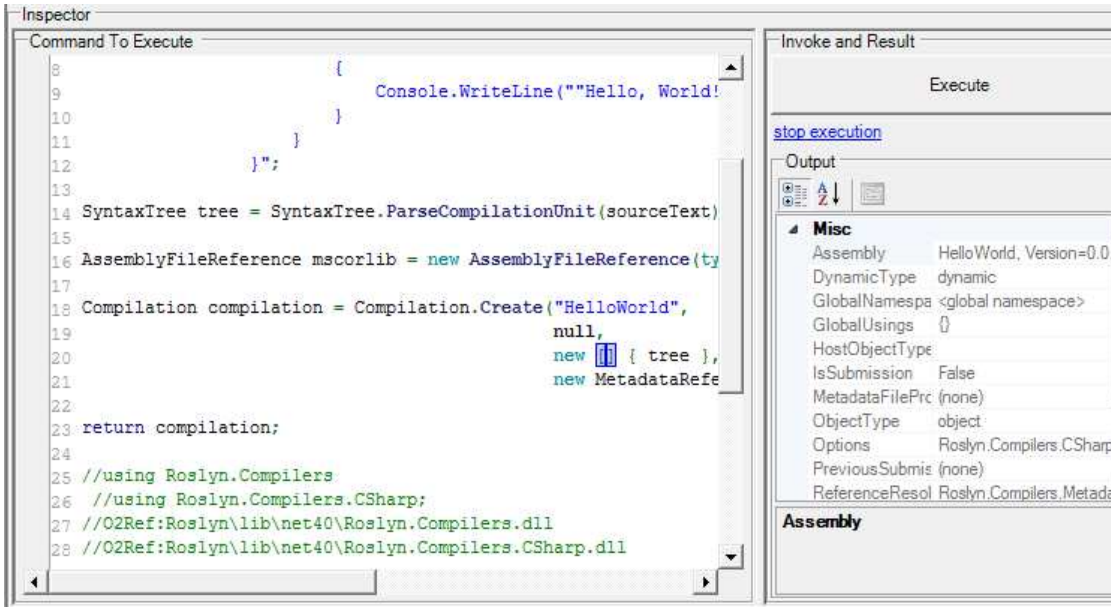
```
SyntaxTree tree = SyntaxTree.ParseCompilationUnit(
                @"using System;

                namespace HelloWorld
                {
                    class Program
                    {
                        static void Main(string[] args)
                        {
                            Console.WriteLine(""Hello, World!"");
                        }
                    }
                }");

NamespaceDeclarationSyntax oldNamespace = tree.Root.DescendentNodes()
                          .OfType<NamespaceDeclarationSyntax>()
                          .First();

NameSyntax namespaceName = Syntax.ParseName("Roslyn.Compilers");

NamespaceDeclarationSyntax newNamespace =
                    oldNamespace.Update(oldNamespace.NamespaceKeyword,
                                        namespaceName,
                                        oldNamespace.OpenBraceToken,
                                        oldNamespace.Externs,
                                        oldNamespace.Usings,
                                        oldNamespace.Members,
                                        oldNamespace.CloseBraceToken,
                                        oldNamespace.SemicolonTokenOpt);

//in the doc the (CompilationUnitSyntax) case was missing
var newRoot = (CompilationUnitSyntax)tree.Root.ReplaceNode(oldNamespace,
newNamespace);
//in the doc the tree.FileName value is lowercase (which is wrong)
SyntaxTree newTree = SyntaxTree.Create(tree.FileName, newRoot,tree.Options);
return newTree;

//using Roslyn.Compilers
  //using Roslyn.Compilers.CSharp;
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
```

```
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```

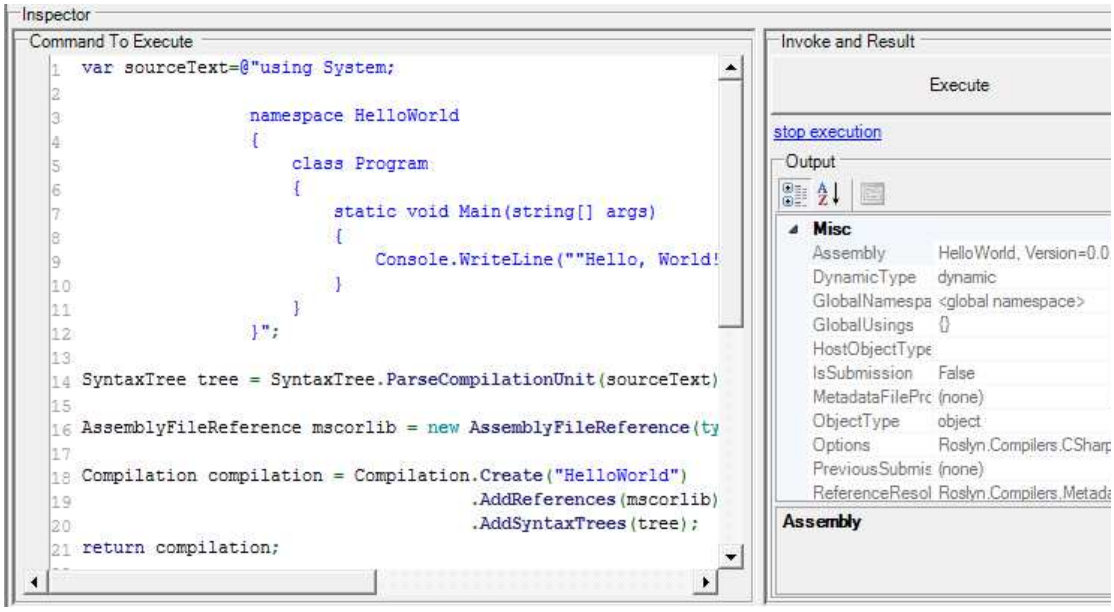


4.2 Obtaining a Compilation

```
var sourceText=@"using System;

                namespace HelloWorld
                {
                    class Program
                    {
                        static void Main(string[] args)
                        {
                            Console.WriteLine(""Hello, World!"");
                        }
                    }
                }";

SyntaxTree tree = SyntaxTree.ParseCompilationUnit(sourceText);

AssemblyFileReference mscorlib = new AssemblyFileReference(typeof
(object).Assembly.Location);

Compilation compilation = Compilation.Create("HelloWorld", null, new [] {
tree }, new MetadataReference[] { mscorlib });

return compilation;

//using Roslyn.Compilers
  //using Roslyn.Compilers.CSharp;
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```

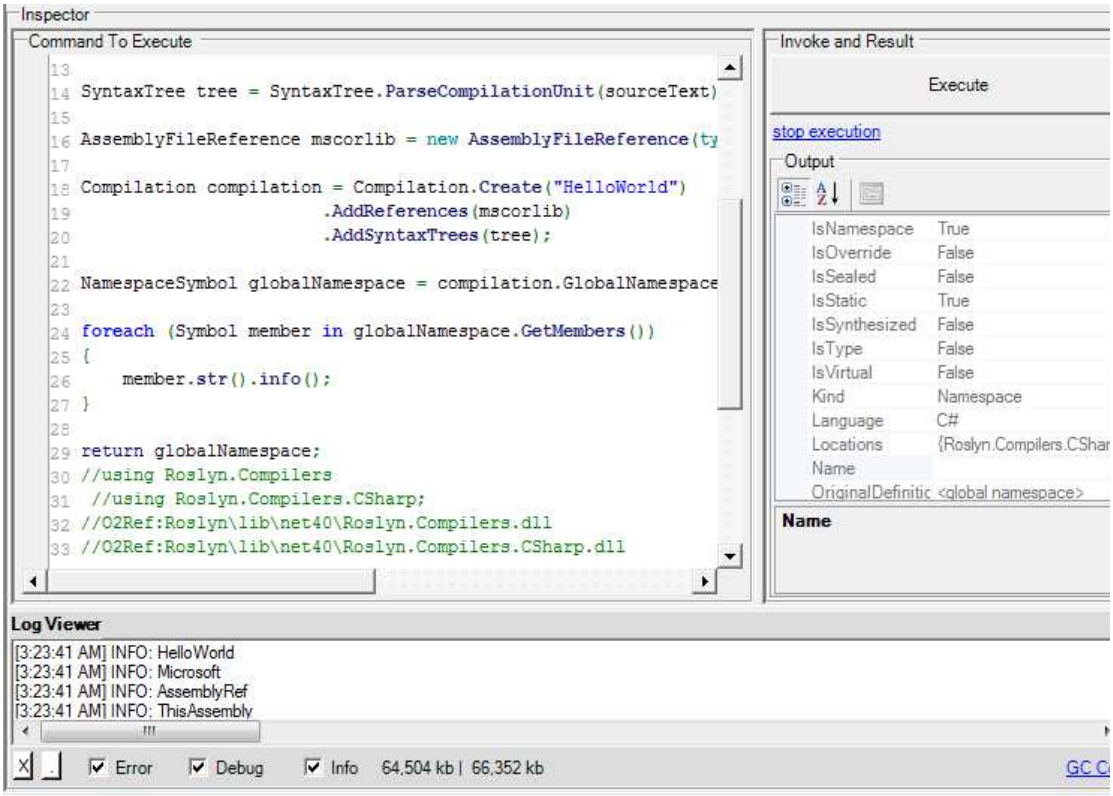



```
var sourceText=@"using System;

                namespace HelloWorld
                {
                    class Program
                    {
                        static void Main(string[] args)
                        {
                            Console.WriteLine(""Hello, World!"");
                        }
                    }
                }";

SyntaxTree tree = SyntaxTree.ParseCompilationUnit(sourceText);

AssemblyFileReference mscorlib = new AssemblyFileReference(typeof
(object).Assembly.Location);

Compilation compilation = Compilation.Create("HelloWorld")
                                     .AddReferences(mscorlib)
                                     .AddSyntaxTrees(tree);
return compilation;

//using Roslyn.Compilers
  //using Roslyn.Compilers.CSharp;
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```



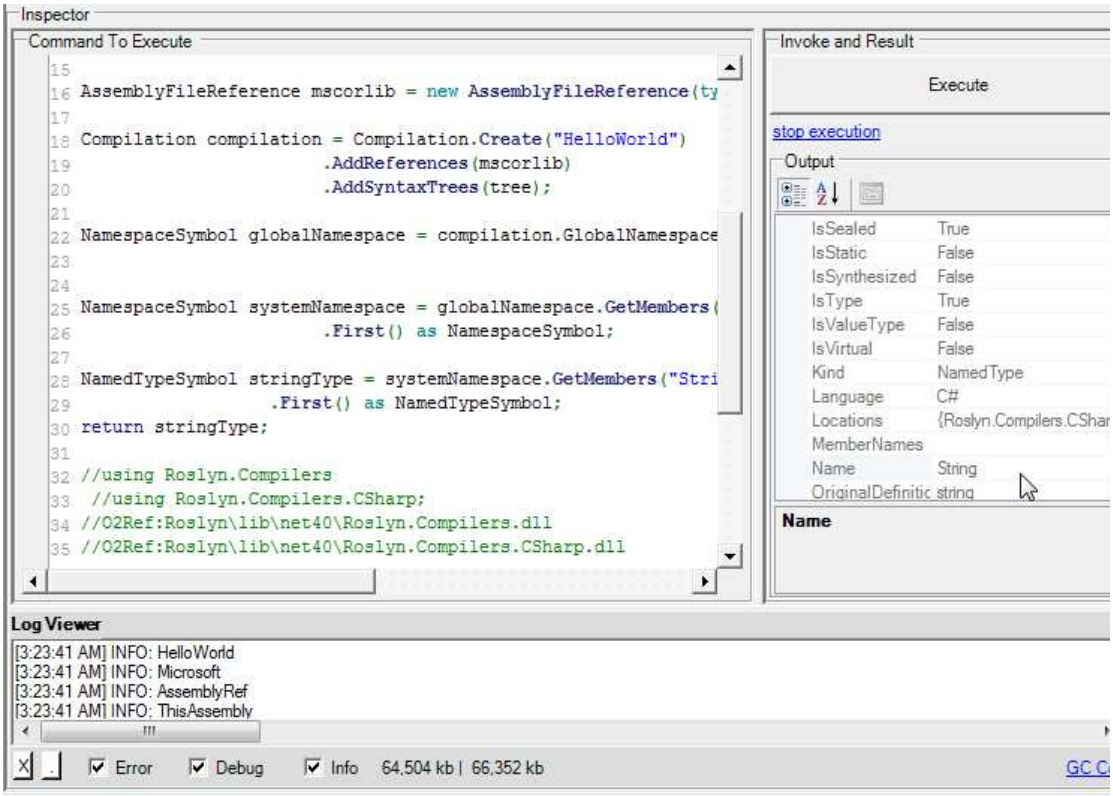
4.3 Obtaining Symbols

```
var sourceText=@"using System;

                namespace HelloWorld
                {
                    class Program
                    {
                        static void Main(string[] args)
                        {
                            Console.WriteLine(""Hello, World!"");
                        }
                    }
                }";

SyntaxTree tree = SyntaxTree.ParseCompilationUnit(sourceText);

AssemblyFileReference mscorlib = new AssemblyFileReference(typeof
(object).Assembly.Location);

Compilation compilation = Compilation.Create("HelloWorld")
                                     .AddReferences(mscorlib)
                                     .AddSyntaxTrees(tree);

NamespaceSymbol globalNamespace = compilation.GlobalNamespace;

foreach (Symbol member in globalNamespace.GetMembers())
{
       member.str().info();
}

return globalNamespace;
//using Roslyn.Compilers
  //using Roslyn.Compilers.CSharp;
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```



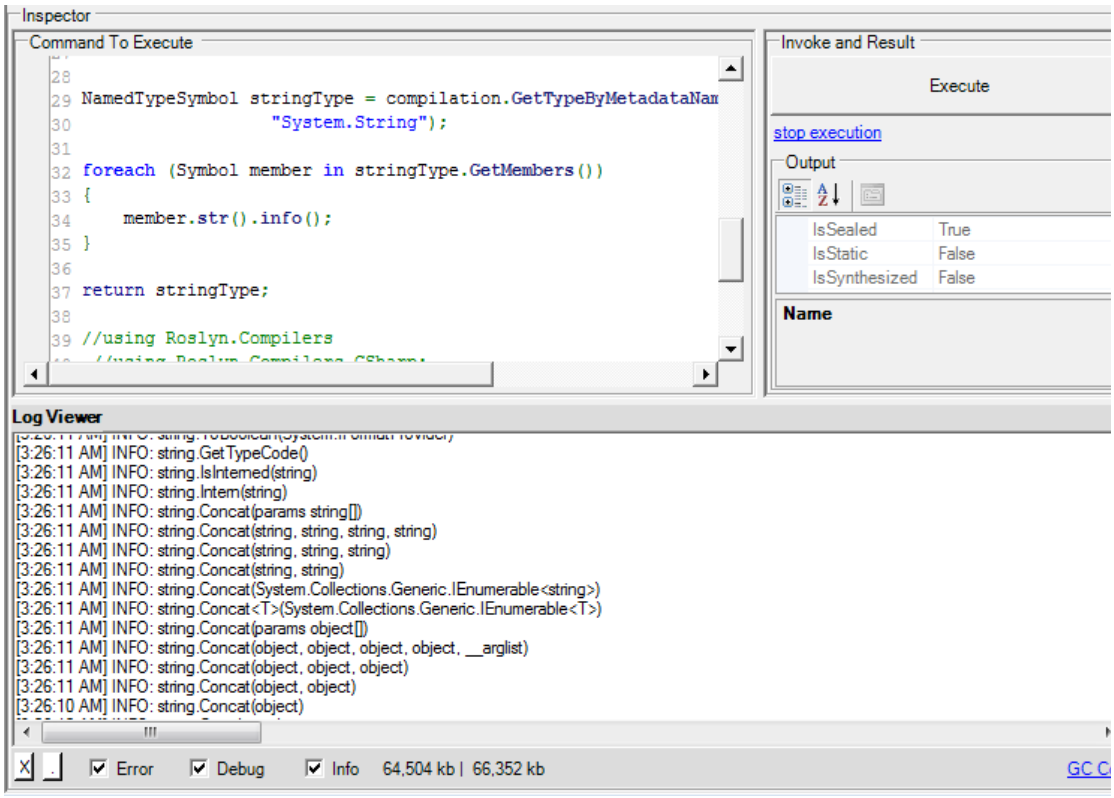

```
var sourceText=@"using System;

                namespace HelloWorld
                {
                    class Program
                    {
                        static void Main(string[] args)
                        {
                            Console.WriteLine(""Hello, World!"");
                        }
                    }
                }";

SyntaxTree tree = SyntaxTree.ParseCompilationUnit(sourceText);

AssemblyFileReference mscorlib = new AssemblyFileReference(typeof
(object).Assembly.Location);

Compilation compilation = Compilation.Create("HelloWorld")
                                     .AddReferences(mscorlib)
                                     .AddSyntaxTrees(tree);

NamespaceSymbol globalNamespace = compilation.GlobalNamespace;


NamespaceSymbol systemNamespace = globalNamespace.GetMembers("System")
                                  .First() as NamespaceSymbol;

NamedTypeSymbol stringType = systemNamespace.GetMembers("String")
                             .First() as NamedTypeSymbol;
return stringType;

//using Roslyn.Compilers
  //using Roslyn.Compilers.CSharp;
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
```

```
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```



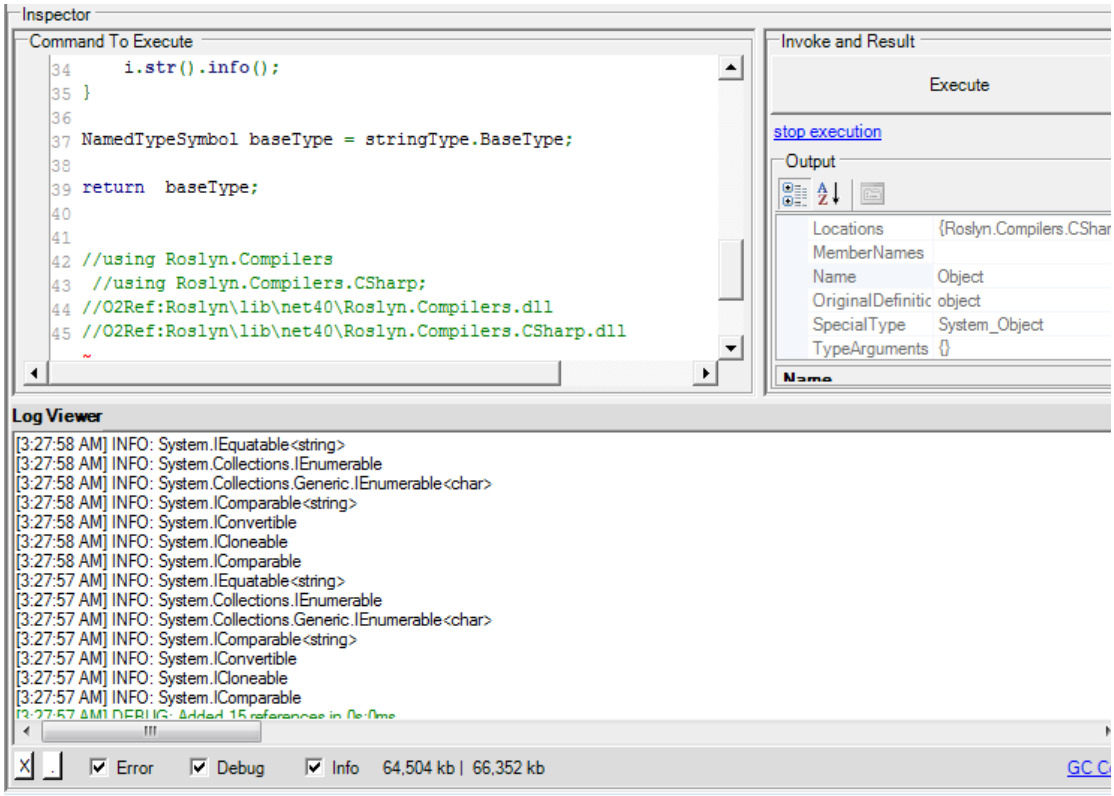

```
var sourceText=@"using System;

                namespace HelloWorld
                {
                    class Program
                    {
                        static void Main(string[] args)
                        {
                            Console.WriteLine(""Hello, World!"");
                        }
                    }
                }";

SyntaxTree tree = SyntaxTree.ParseCompilationUnit(sourceText);

AssemblyFileReference mscorlib = new AssemblyFileReference(typeof
(object).Assembly.Location);

Compilation compilation = Compilation.Create("HelloWorld")
                                     .AddReferences(mscorlib)
                                     .AddSyntaxTrees(tree);

NamespaceSymbol globalNamespace = compilation.GlobalNamespace;


NamespaceSymbol systemNamespace = globalNamespace.GetMembers("System")
                                  .First() as NamespaceSymbol;


NamedTypeSymbol stringType = compilation.GetTypeByMetadataName(
                             "System.String");

foreach (Symbol member in stringType.GetMembers())
{
```

```
    member.str().info();
}

return stringType;

//using Roslyn.Compilers
  //using Roslyn.Compilers.CSharp;
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```




```
var sourceText=@"using System;

                namespace HelloWorld
                {
                    class Program
                    {
                        static void Main(string[] args)
                        {
                            Console.WriteLine(""Hello, World!"");
                        }
                    }
                }";

SyntaxTree tree = SyntaxTree.ParseCompilationUnit(sourceText);

AssemblyFileReference mscorlib = new AssemblyFileReference(typeof
(object).Assembly.Location);

Compilation compilation = Compilation.Create("HelloWorld")
                                     .AddReferences(mscorlib)
                                     .AddSyntaxTrees(tree);

NamespaceSymbol globalNamespace = compilation.GlobalNamespace;
```

```
NamespaceSymbol systemNamespace = globalNamespace.GetMembers("System")
                                  .First() as NamespaceSymbol;


NamedTypeSymbol stringType = compilation.GetTypeByMetadataName(
                             "System.String");

foreach (NamedTypeSymbol i in stringType.Interfaces)
{
    i.str().info();
}

NamedTypeSymbol baseType = stringType.BaseType;

return       baseType;


//using Roslyn.Compilers
//using Roslyn.Compilers.CSharp;
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```



4.4 Asking Semantic Questions

```
var sourceText=@"using System;

                namespace HelloWorld
                {
                    class Program
                    {
                        static void Main(string[] args)
                        {
                            Console.WriteLine(""Hello, World!"");
```

```
                    }
                }
            }";

SyntaxTree tree = SyntaxTree.ParseCompilationUnit(sourceText);

AssemblyFileReference mscorlib = new AssemblyFileReference(typeof
(object).Assembly.Location);

Compilation compilation = Compilation.Create("HelloWorld")
                                     .AddReferences(mscorlib)
                                     .AddSyntaxTrees(tree);


SemanticModel semanticModel = compilation.GetSemanticModel(tree);

return       semanticModel;


//using Roslyn.Compilers
  //using Roslyn.Compilers.CSharp;
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.dll
//O2Ref:Roslyn\lib\net40\Roslyn.Compilers.CSharp.dll
```

-
-
-
-
-
-

.

.

.

.