



➤ Microsoft .NET

➤ Writing .NET Classes

## Writing .NET Classes - Example

This example builds an Assembly called APLClasses1.dll which contains a .NET Namespace called APLClasses. This will contain a single .NET Class called Primitives that exports a single method called IndexGen.

### Making the Class

First we create a container namespace `#.APLClasses` that will represent the .NET Namespace in the assembly:

```
clear ws
      )NS APLClasses
#.APLClasses
```

Next, using `□WC`, we create a NetType object called `#.APLClasses.Primitives`. Note that the default BaseClass for a NetType object is System.Object.

```
      )CS APLClasses
#.APLClasses
      'Primitives' □WC 'NetType'
```

Then, inside the Primitives namespace, we set `□USING` so that the code inside this object can reference the .NET base types such as Int32.

```
      )CS Primitives
#.APLClasses.Primitives
      □USING←'System'
```

Next, we write the `Primitives.IndexGen` function. As we will see later, it is not necessary for a function to have the same name as the exported method that it implements, but it is the default.

```
      ▽ R←IndexGen N
[ 1 ] R←ιN
      ▽
```

#### Dyalog Ltd

South Barn  
Minchens Court  
Minchens Lane  
Bramley  
Hampshire  
RG26 5BH  
United Kingdom

#### Phone:

+ 44 (0) 1256 830 030

#### Fax:

+ 44 (0) 1256 830 031

#### e-mail:

sales@dyalog.com

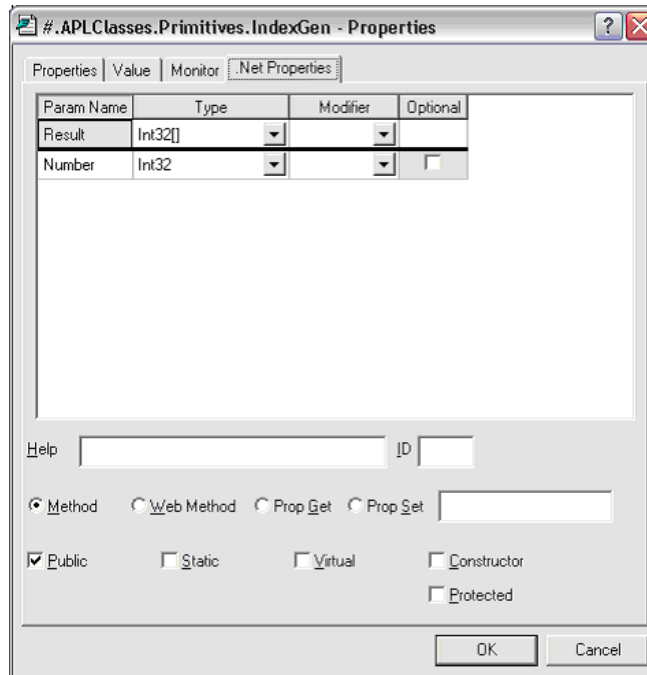


▶ Microsoft .NET

▶ Writing .NET Classes

### Defining the IndexGen Method

The next step is to define the public characteristics for the exported method . This is done using the .Net Properties page of the Properties dialog box for the *IndexGen* function as shown below.



1. To make the function available to a client application, check the Public check box.
2. To export the function as a method (as opposed to a Web Method, or a Property Get/Set function), select the Method radio button.
3. Enter Int32[] in the box for Result Type. This says that IndexGen returns an array of integers.
4. Enter Int32 in the box for the Param1 Type and (optionally) rename the parameter Param1, in this case, to "number".
5. Click OK.

Note that APL will at this stage check the data types you have specified for the result and for the method's parameters. If one or more of the data types are not recognised as available .NET Types (Classes), you will be informed by a message box. If you see such a warning you have either entered an incorrect data type, or you have not set `USING` correctly. In this case, the only data type used is Int32, which is a Type, defined in the .NET Namespace System, and `USING` is set to `'System'`, so all will be well. The next step is not strictly necessary, but it does make good sense to `)SAVE` the workspace at this stage. The name you choose for the workspace will be the default name for the assembly

```

)CS
#
)WSID samples\APLClasses\aplclasses1
was CLEAR WS
)SAVE
samples\APLClasses\aplclasses1 saved Wed No
v 21 12:30:38 2001
    
```

#### Dyalog Ltd

South Barn  
Minchens Court  
Minchens Lane  
Bramley  
Hampshire  
RG26 5BH  
United Kingdom

**Phone:**

+ 44 (0) 1256 830 030

**Fax:**

+ 44 (0) 1256 830 031

**e-mail:**

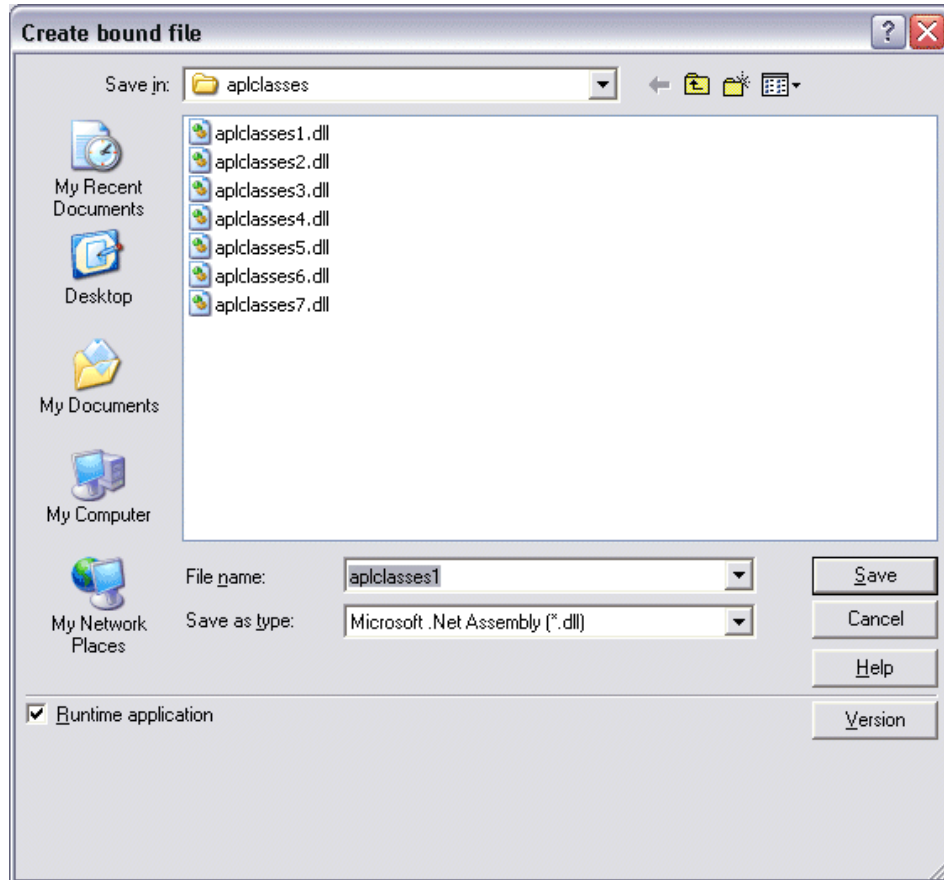
sales@dyalog.com



### Exporting the Class

Now you are ready to create the assembly. This is done by selecting Export... from the Session File menu. This displays the following dialog box.

- Microsoft .NET
- Writing .NET Classes



This gives you the opportunity to change the name or path of the assembly. The *Runtime application* checkbox allows you to choose to which if the two versions of the Dyalog dynamic link library the assembly will be bound.

Finally click Save.

#### Dyalog Ltd

South Barn  
Minchens Court  
Minchens Lane  
Bramley  
Hampshire  
RG26 5BH  
United Kingdom

**Phone:**

+ 44 (0) 1256 830 030

**Fax:**

+ 44 (0) 1256 830 031

**e-mail:**

sales@dyalog.com



➤ Microsoft .NET

➤ Writing .NET Classes

APL now makes the assembly and, as it does so, displays information in the Status window as shown below. If any errors occur during this process, the Status window will inform you.

```

Dyalog APL/W - Status
File Options
Declared Assembly aplclasses1
Declared Module aplclasses1 in file C:\dyalog101\samples\aplclasses\aplclasses1.dll
  Declared Type APLClasses.Primitives
    Compiling Method "IndexGen"
      Parameter type "Int32" resolved to System.Int32
      Result type "Int32[]" resolved to System.Int32[]
    Compiled Method "IndexGen"
  Emitted Type APLClasses.Primitives
Emitted Assembly to file "C:\dyalog101\samples\aplclasses\aplclasses1.dll"
    
```

### The Proof

The following picture shows the Dyalog Class as viewed by the Microsoft .NET Class browser, ildasm.exe. As you can see, the IndexGen method is exported as a method that takes a single Int32 parameter and returns an array of type Int32. This class is now ready for use by any other .NET compliant language.

```

C:\dyalog101\samples\aplclasses\aplclasses1.dll - IL DASM
File View Help
C:\dyalog101\samples\aplclasses\aplclasses1.dll
├── MANIFEST
├── APLClasses
│   └── Primitives
│       ├── .class public auto ansi
│       ├── $ToDyalog : private static class [bridge11]ToDyalog
│       ├── $idx : private int32
│       ├── .cctor : void()
│       ├── .ctor : void()
│       ├── BaseConstructor : void()
│       └── IndexGen : int32[(int32)]
    
```

**Dyalog Ltd**  
 South Barn  
 Minchens Court  
 Minchens Lane  
 Bramley  
 Hampshire  
 RG26 5BH  
 United Kingdom

**Phone:**  
 + 44 (0) 1256 830 030

**Fax:**  
 + 44 (0) 1256 830 031

**e-mail:**  
 sales@dyalog.com