3

Programming with COM

Exercise 3A: Program with COM (VB.NET)
Estimated time: 45 minutes
Challenge: Zoom to full extent
Estimated time: 2 minutes

Exercise 3B: Program with COM (C#)
Estimated time: 45 minutes
Challenge: Zoom to full extent
Estimated time: 2 minutes

Exercise 3B: Program with COM (C#)

Estimated time: 45 minutes

In this exercise, you will access the properties of a map in a map document.

In this exercise, you will:

- Use developer resources
- Access the properties of a map
- List the values for the properties

Exercise shortcut

1. Start Visual Studio and create a new Windows application.
2. Add a button to your form.
3. Write code to create a new map document object. Enter an .mxd file as input.
4. Return the first map in your map document and write its name and one or two other of its properties to the Output window.
5. Write the name of the active view's focus map to the Output window. Write the minimum and maximum X and Y properties of the active view's extent to the Output window.

Step 1: Create a new project in Visual Studio

☐ Start Visual Studio.

☐ Create a new project, using the following as a guide:

Property	Value
Language:	Visual C#
Template:	Windows Application
Name:	**MapDocInfo**
Location:	**\Student\IPAN\Exercise03**
Solution name:	MapDocInfo

Make sure the Create directory for solution check box is checked.

Your project opens in design view to a Visual Studio form control. In the Solution Explorer, you see that your project contains one folder, two files, and project references.

Next, you will direct output to the Output window.

☐ From the Tools menu, choose Options.

☐ On the left of the Options dialog box, select Debugging.

☐ Make sure the Redirect all Output Window text to the Immediate Window check box is unchecked.

☐ Click OK to close the dialog box.

Step 2: Add a custom button to your project

Now, to prepare your project to retrieve some information from your map, you will add a Visual Studio button.

☐ If necessary, display the Toolbox window. *Tip:* Click the Toolbox button  .

☐ Drag a Button control onto your form from the Common Controls group of the Toolbox window.

☐ In the Properties window, change the following properties for your button:

- (Name): **btnInfo**
- Text: **Info**

Step 3: Add code to your custom button

In this step, you will access a map document and determine its name and other properties—without viewing the data. Instead, you will rely on other resources, such as the .NET Help for VS2005, the code completer, and the lecture slides.

☐ Double-click your Info button to open its click event.

☐ Declare the variable `mapDoc` to hold a reference to `IMapDocument`.

☐ From the Build menu, choose Build MapDocInfo to build your project.

☐ Hover your mouse pointer over `IMapDocument`.

Your syntax is not recognized, so you will need to add the appropriate reference. You will use the Library Locator to determine which reference you need.

☐ Open Windows Explorer.

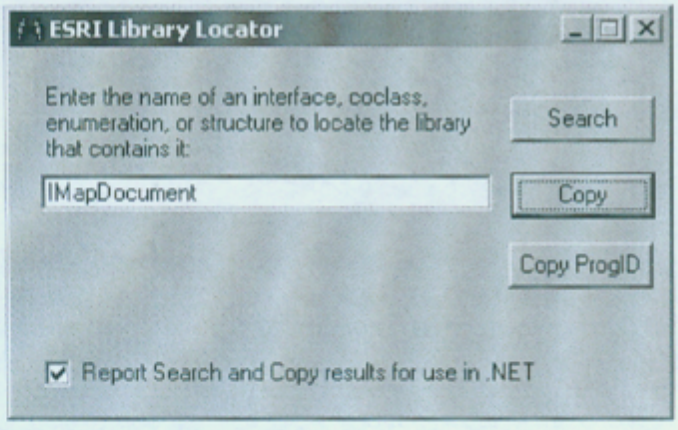
☐ Navigate to C:\Program Files\ArcGIS\DeveloperKit\Tools.

☐ Right-click LibraryLocator.exe and choose Send To > Desktop (create shortcut).

☐ Close Windows Explorer.

☐ Open the Library Locator using the new shortcut.

☐ Enter **IMapDocument**, then check the Report Search and Copy results for use in .NET check box.



☐ Click Search.

Question 1: Which library do you need to add to your project?

_____

☐ Click Copy.

☐ In Visual Studio, from the Project menu, choose Add ArcGIS Reference.

Because you will not be adding any ArcGIS Engine controls to your application, you can use ArcGIS Desktop references and libraries.

☐ On the Add ArcGIS Reference dialog box, on the References tab, expand Desktop ArcMap.

☐ Double-click ESRI.ArcGIS.Carto to add it to the list of selected assemblies, then click Finish.

☐ Near the top of your code, above your namespace, type **`using`**, then right-click and choose Paste to add the assembly reference to your project.

☐ Build your project.

☐ Click inside your `btnInfo_Click` procedure.

Question 2: What changed in the variable declaration you wrote?

_____

_____

The `IMapDocument` interface is found on the `MapDocument` coclass, which makes your `mapDoc` variable a creatable object.

☐ Use the following code to create a new **`MapDocument`** object. You will learn more about this syntax in the next lesson.

```
IMapDocument mapDoc = new MapDocument();
```

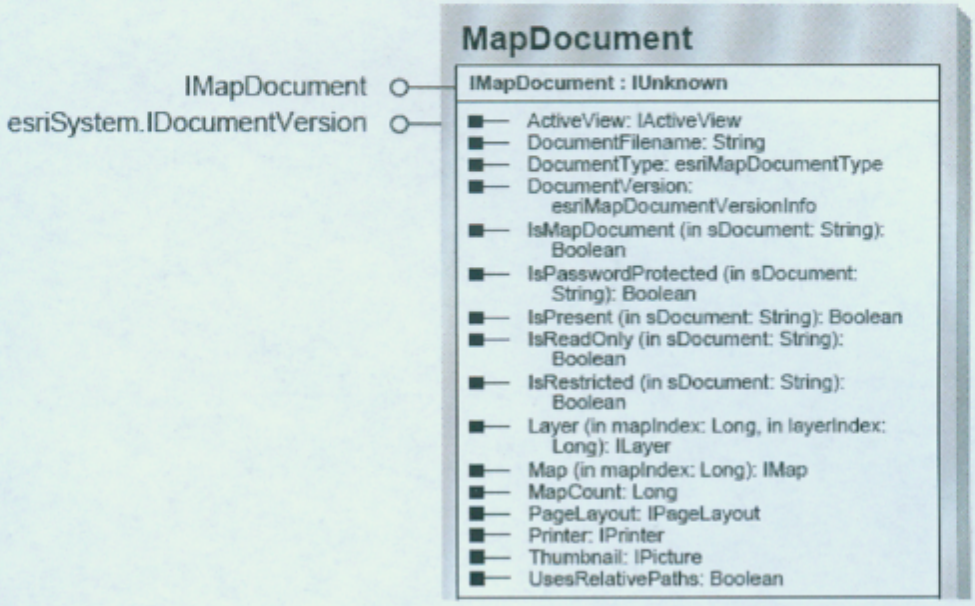
☐ Build your project.

> **! For the remainder of this exercise and the remaining exercises, remember to build your project (or solution) as needed. It is especially important to build your project after you have corrected syntax errors.**

Now `MapDocument` displays an error.

☐ To eliminate the new error, add an ArcGIS reference to ESRI.ArcGIS.System the same way you added a reference to ESRI.ArcGIS.Carto. You do not need a using statement.

Many assemblies have the System namespace as a dependency. In the following graphic, notice that the MapDocument coclass has two interfaces.



Although you are working only with IMapDocument, IDocumentVersion is found in the System assembly; thus, your project needs a reference to the System assembly.

There are different types of map documents, one of which is an .mxd file. Next, you will access an .mxd file that installed with your data.

☐ Write the following code to open a map document (use the complete path, including drive letter):

```
mapDoc.Open("\\Student\\IPAN\\Exercise03\\SouthAmerica.mxd", "");
```

☐ Open the .NET Help for VS2005.

☐ Click Search to display the Search tab.

☐ In the Search box, type **IMapDocument interface**, then click Search.

☐ Click Local Help on the right of the tab, then click the IMapDocument Interface link on the left.

☐ Locate the property that will tell you the type of map document currently loaded in the object, then explore its links.

Question 3: What parameter does the property return? How many types of documents does it include?

_____

☐ In Visual Studio, make sure your Output window is displayed. If you don't see it, from the View menu, choose Output.

☐ After the code that opens the map document, complete the following code to write the value of the property to the Output window:

```
System.Diagnostics.Debug.WriteLine(mapDoc._____);
```

☐ Save your project.

☐ Build your project.

☐ Run your project  .

☐ If necessary, display the Output window.

☐ Click the Info button on your application.

Question 4: What value is returned in the Output window when you click the Info button?

_____

☐ Close your application.

☐ In your button's click event, skip five or six lines after `Debug.WriteLine`, then use the following syntax to close your application as soon as the process finishes:

```
this.Close()
```

☐ Run your project and test the change.

Step 4: Work with the map

Even though your application currently does not display a map, you can still access the map and the active view for your map document. In this step, you will use .NET casting to move between the interfaces for the map and the active view.

☐ *Before* the code that closes the application, declare the variable **`inMap`** to hold a reference to **`IMap`**.

☐ Complete the following code to set `inMap` equal to the first map in your map document:

```
inMap = _____.get_map(0);
```

☐ Write the name of the map to the Output window.

☐ Run your project and test your code.

☐ After the application closes, click the Clear All button  in your Output window to eliminate clutter.

☐ Use the developer resource of your choice to find additional properties for your map.

☐ Find the property for the map's distance units.

☐ Write the value for this property to the Output window.

☐ If you would like, write the value of one or more other properties to the Output window.

> **Beware of non-string and non-numeric properties, as they may cause runtime errors.**

☐ Build and run your project.

☐ Test your changes.

Question 5: What are the map's distance units?

Step 5: Work with the active view

Every map has an extent. In this step, you will find the X and Y values for the lower-left and upper-right corners of the extent of your map. You will add your code to your custom button.

☐ Declare the variable **`actView`** as **`IActiveView`**.

☐ Write code to cast (perform a COM QueryInterface) from **`IMap`** to **`IActiveView`**. (Hint: For help with the syntax, refer to the lecture slides.)

☐ Write the name of the focus map to the Output window.

☐ Test your code.

Notice that the name of the active view's focus map is the same as the name of the map you are accessing from your map document. The variables `inMap` and `actView` are pointing to the same map.

☐ Write the **`Extent`** property of the active view to the Output window.

☐ Build your project.

There is an error on the `Extent` property of the active view.

☐ Hover your mouse pointer over the error.

Question 6: Which assembly is required?

_____

☐ Add an ArcGIS reference to the required assembly.

☐ At the top of your code, import the assembly to your project with the following code:

```
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS._____;

namespace MapDocInfo
{
    public partial class Form1 : Form
```

☐ Modify your code to write the minimum X value of the extent. Include a descriptive label in the argument for `Debug.WriteLine`.

☐ Clear the contents of the Output window.

☐ Test your code.

Question 7: What is the minimum X value (rounded to the nearest integer)?

_____

☐ Revise your code to write the minimum X and Y and the maximum X and Y. *Tip:* Use the `Round` function in the Math library to round to zero decimal places.

☐ Test your code.

Question 8: What is the maximum Y?

You now have an optional step and a challenge step you can perform:

- Optional step 6 shows you how to load a map document into a MapControl on your application
- The challenge step lets you write code to display your application's map at full extent

☐ If you would like to perform one or both of these steps, do so now. Otherwise, save your project and close Visual Studio.

Step 6: (Optional) Add a MapControl to your project

In this step, you will add your map document to a MapControl.

☐ Click the View Designer button  at the top of the Solution Explorer to view your project in design view.

☐ Drag a MapControl onto your form from the ArcGIS Windows Forms group of the Toolbox window.

☐ Resize your form, if necessary.

☐ Run your project.

A license error appears.

The custom application you are writing uses ArcObjects to perform a GIS task. ArcMap and ArcCatalog also use ArcObjects to perform GIS tasks. When you start ArcMap, for example, there is some processing that goes on behind the scenes to check for a license. When you run your custom application, you become responsible for checking for a license. In order to use the MapControl, you will need to include license handling in your project. In the previous exercise, you added the LicenseControl. This time, you will add code that has already been written.

☐ Close the error message and close your application.

☐ From the Project menu, choose Add ArcGIS License Checking.

☐ In the ArcGIS License Initializer dialog box, double-click ArcView in the Products list to select it.

☐ Check the Shut down this application if any selected license(s) is not available check box.

☐ Click OK.

Code that manages the license is added to your project.

Question 9: Which module initializes the license with the application and shuts down the license?

_____

The license-handling routine added references and namespaces that it needs, but you need to add a using statement to the code for your form.

☐ Add the following code after the other using statements:

```
using ESRI.ArcGIS.esriSystem;
```

Next, you will load your map document into the MapControl.

☐ View the Form1 code.

☐ Just before the line `this.Close()`, complete the following code to load the same map document that you have been exploring:

```
AxMapControl1.LoadMxFile(mapDoc._____)
```

☐ Comment the line `this.Close()`.

☐ Test your code.

SouthAmerica.mxd displays in the MapControl.

☐ Close your application.

☐ If you would like to write code to display your application's map at full extent, go on to the challenge step. Otherwise, save your project and close Visual Studio.

Conclusion

In this exercise, you accessed the properties of an existing map without viewing it. In your code, you moved between two interfaces on the same object by casting, the .NET technique for the process known as QueryInterface (QI) in the world of COM.

Challenge: Zoom to full extent

Estimated time: 2 minutes

☐ In one line of code, set the map's extent equal to its full extent.

Answers to Exercise 3B Questions

Question 1: Which library do you need to add to your project?

Answer: ESRI.ArcGIS.Carto

Question 2: What changed in the variable declaration you wrote?

Answer: `IMapDocument` no longer displays an error, but the variable `mapDoc` is flagged as never used

Question 3: What parameter does the property return? How many types of documents does it include?

Answer: esriMapDocumentType; four

Question 4: What value is returned in the Output window when you click the Info button?

Answer: esriMapDocumentTypeMxd

Question 5: What are the map's distance units?

Answer: esriMiles

Question 6: Which assembly is required?

Answer: ESRI.ArcGIS.Geometry

Question 7: What is the minimum X value (rounded to the nearest integer)?

Answer: -77 89

Question 8: What is the maximum Y?

Answer: 13 -7

Question 9: Which module initializes the license with the application and shuts down the license?

Answer: LicenseInitializer

Exercise Solution

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Geometry;

namespace MapDocInfo
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnInfo_Click(object sender, EventArgs e)
        {
            IMapDocument mapDoc = new MapDocument();
            mapDoc.Open("\\Student\\IPAN\\Exercise03\\SouthAmerica.mxd", "");
            System.Diagnostics.Debug.WriteLine(mapDoc.DocumentType);

            IMap inMap;
            inMap = mapDoc.get_Map(0);
            System.Diagnostics.Debug.WriteLine("Map name: " + inMap.Name);

            System.Diagnostics.Debug.WriteLine(
               "Distance units: " + inMap.DistanceUnits);

            IActiveView actView;
            actView = inMap as IActiveView;

            System.Diagnostics.Debug.WriteLine(
               "Focus map name: " + actView.FocusMap.Name);

            System.Diagnostics.Debug.WriteLine("MinX: " + actView.Extent.XMin);

            System.Diagnostics.Debug.WriteLine(
                "MinX: " + Math.Round(actView.Extent.XMin, 0)  + " " +
                "MinY: " + Math.Round(actView.Extent.YMin, 0) + " " +
                "MaxX: " + Math.Round(actView.Extent.XMax, 0) + " " +
                "MaxY: " + Math.Round(actView.Extent.YMax, 0));

            System.Diagnostics.Debug.WriteLine(mapDoc.DocumentFilename);
            axMapControl1.LoadMxFile(mapDoc.DocumentFilename);

            //this.Close();
        }
    }
}
```

Challenge Solution: Zoom to full extent

```
axMapControl1.Extent = axMapControl1.FullExtent;
```