



Echoview Tutorial: Introduction to COM Scripting

Contents

Overview	5
Prerequisites	5
Echoview modules.....	5
Contacting Echoview.....	5
Set up	5
Troubleshooting.....	5
Topic 1: Echoview COM.....	6
Why script with Echoview?	6
How do you do it?	7
Actions in the script.....	7
Anatomy of a script (VBScript).....	7
Grammar and language	9
Structure.....	9
COM Exec ()	11
Example script contrasting COM approaches	11
<i>Tips:</i>	12
What else do you need?	12
<i>Tips:</i>	12
Topic 2: COM Concepts and Terminology	14
What is COM scripting?	14
Basic concepts and terminology	14
COM objects and classes	14
A COM object model	15
COM properties.....	15
COM methods, parameters and return values.....	15
Topic 2 exercises.....	17
Further reading	18
Topic 3: Echoview COM Basics	19
Register Echoview COM.....	19
The Echoview Component Object Model (COM).....	19
The EvApplication class.....	21
Echoview classes.....	21
<i>Tips:</i>	21
Echoview objects	21
Echoview object properties	22

Echoview object methods	22
Echoview collections	23
Overview of Echoview COM objects	24
Echoview variables	24
Acoustic variables	25
Regions and region classes	25
Lines	26
Filesets and data files	26
EV file properties	26
Topic 3 exercises	29
Topic 4: Basic VBScript Concepts	31
VBScript syntax	31
Accessing properties and methods	31
Brackets and parentheses	31
Using comments	31
Variables and keywords	31
Keywords	31
Script variables	31
Using variables in a script	32
Declaring an object type variable	32
Using Option Explicit	33
Declaring a simple type variable	33
Scripting procedures	35
Sub procedures (AKA procedures)	36
Function	36
Conditional statements	37
<i>Tips:</i>	39
Syntax highlighting	39
Common VBScript conventions	39
Variable lifetime and where a variable is applied	39
Windows objects and classes	39
Topic 4 exercises	40
Further reading	40
Topic 5: Create a Script for Echoview	41
Interpreting basic scripts	41
Topic 5 exercises	49

Topic 6: Viewing Echoview COM Objects and Creating Scripts	54
Text editor programs designed for editing code	54
Windows Visual Basic editor	56
Opening Microsoft Visual Basic using the Developer tab	56
To write or view scripts – Code window	59
References	60

Overview

This tutorial is optimized for Echoview 15 and provides an introduction to COM Scripting with Echoview.

This tutorial is not intended as a comprehensive user manual.

Further information on Echoview tools and topics can be found in the latest version of the Echoview help file. This can be viewed online and is installed with Echoview. Press F1 when using Echoview to open the help file and read context-sensitive information.

Throughout this tutorial further reading is referred to the Echoview help file or other Echoview learning materials: <https://www.echoview.com/support>

Prerequisites

This tutorial assumes you have Echoview installed, and the following skills and knowledge:

- Familiarity with the basic operation of Echoview. We strongly recommend that you complete the "Getting Started with Echoview" tutorial before beginning this tutorial.
- Familiarity with a supported Microsoft Windows™ operating system. For more information refer to the Echoview Help file page: Computer requirements.
- A basic understanding of echosounding techniques and hydroacoustic surveying. For more information, see texts such as *Fisheries Acoustics* (Simmonds and MacLennan, 2005, Blackwell Science, Oxford.).
- Knowledge about the Echoview COM interface (see Help file: Getting started with scripting). Basic Echoview COM information is included in the tutorial.
- Awareness of the workflow for Echoview tasks.
- Knowledge about the structure of a script.
- Knowledge about the syntax and grammar of a script language.

Echoview modules

This tutorial requires a license with the Essentials and Automation modules.

If you do not yet have access to an Echoview license with these modules, please contact info@echoview.com to request an evaluation license. Solution files are also included in the tutorial package to allow you to see the results of module-protected capabilities.

Contacting Echoview

For assistance with this tutorial please contact support@echoview.com.

Set up

We recommend extracting or copying the tutorial files to `C:\Echoview Software\Tutorials\`. If the files are not in this folder, use Windows Explorer to search for them. If they are not loaded on your machine, download and reinstall the tutorial from www.echoview.com or from the Echoview USB drive.

Troubleshooting

If you receive a message saying that the version of Echoview you are running cannot read the file you have opened, you may be running an old version of Echoview. You can download the latest version of Echoview from www.echoview.com.

Topic 1: Echoview COM

This overview briefly touches on the why, what and how of scripting for Echoview. The tutorial topics that follow offer details.

Why script with Echoview?

Scripting allows you to automate tasks in Echoview. Automation can save you processing and analysis time – time that you can devote to complex analysis and result synthesis.

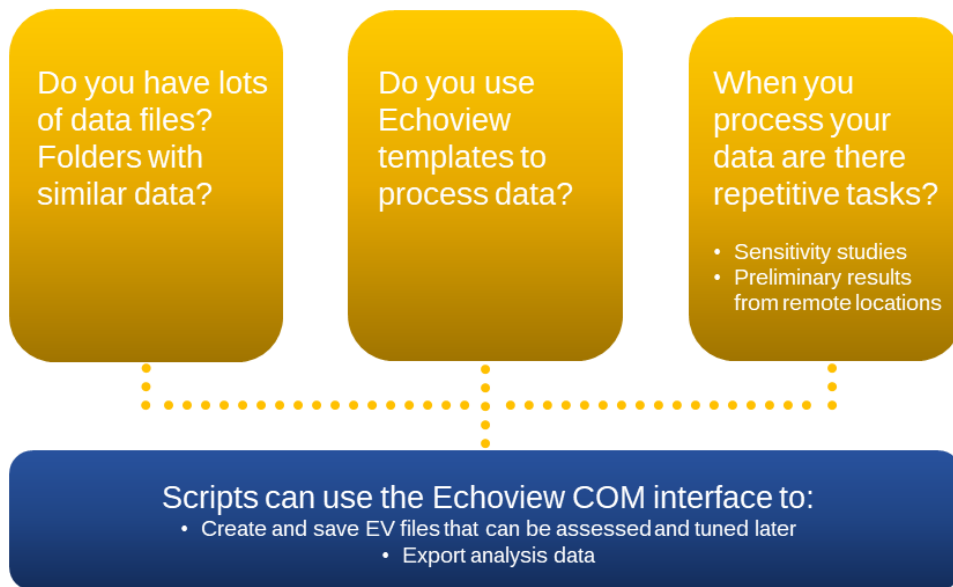


Figure 1. Why script with Echoview?

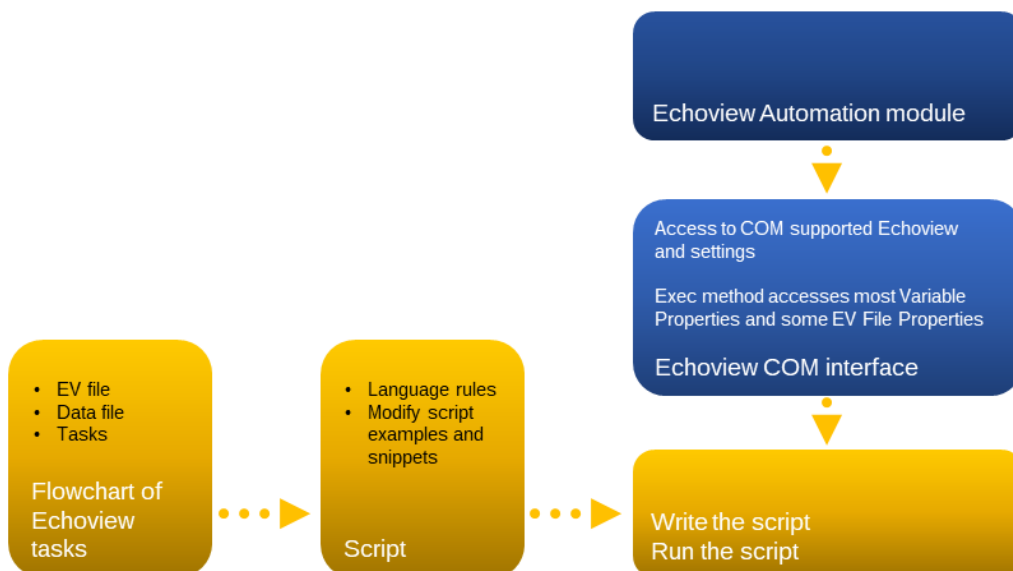


Figure 2. What do you need to start scripting with Echoview?

How do you do it?

This exercise looks at a Visual Basic script written in Notepad.

- It uses the Echoview COM interface.
 - It also uses the Windows VBA¹ interface.
1. In the IntroToCOMScripting tutorial folder, open the .vbs file `Connect to Echoview.vbs` with Windows Notepad. Script files can be written using any plain text editor. Some text editors with coding handling features can help you to construct a script (color coding, COM library browser etc.).

Actions in the script

The script:

- connects the script to Echoview.
- writes to the Echoview Scripting log file.
- displays a message with Echoview version, and Windows Time and Date information.
- disconnects Echoview from the script (and Window script host).
- the text preceded by an apostrophe is commented text. Such text can be used to explain parts of the script or they can be used to inactivate parts of the script code.

Anatomy of a script (VBScript)

```
'The first line below defines (or "dimensions") the variable "EvApp". A COM variable is not the same as
an Echoview variable.

'EvApp allows us to refer to an object in shorthand, like we might use a nickname to refer to a person
Dim EvApp

'This sets EvApp to be the name of the "Object" which is the EV application. This connects the script
to Echoview.

'This is the highest level object in the "Echoview object model" (See Echoview object model diagram)

Set EvApp = CreateObject("EchoviewCom.EvApplication")

'In the next line we have used the syntax ".Version" to get the version property of the EvApp object.
'We have also use Date and Time with the Windows system - not from Echoview
EvApp.Log("write to log")
MsgBox "You connected to Echoview version: " & EvApp.Version & " at " & Date & " " & Time

'In the final line we have quit or closed the connection between the Windows scripting host and
Echoview
EvApp.Quit
```

Figure 3. Connect to Echoview.vbs displayed in Notepad.

```
Dim EvApp
Set EvApp = CreateObject("EchoviewCom.EvApplication")
EvApp.Log("write to log")
MsgBox "You connected to Echoview version: " & EvApp.Version & " at " & Date & " " & Time
EvApp.Quit
```

Figure 4. Parts of the script color-coded with commented text removed

1. Double-click the file to run the script.

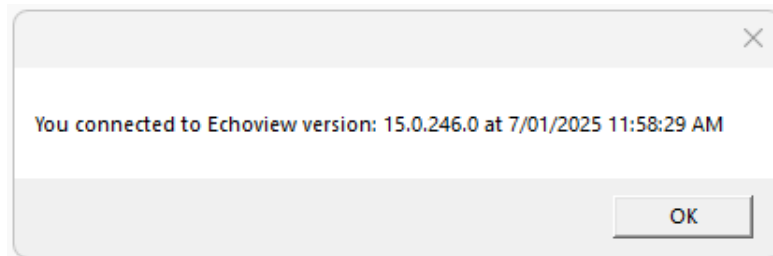


Figure 5. The Windows Message box displays your Echoview version and the date and time.

Grammar and language

The script is a *.vbs file. The language used is VBScript², which is bundled with Microsoft Windows.

VBScript item	Description
EvApp	EvApp is a script variable name that represents the Echoview interface.
Dim	VBScript name that means dimension (or declare) a variable.
Set var_name =	VBScript name that means set var_name to whatever follows the equal sign.
"text"	Used to denote a string of text for a COM interface action. e.g., EvApp.Log("write to log")
&	Used to join parts of the message's text and output. e.g., "You connected to Echoview version: " & EvApp.Version & " at " & Date & " " & Time

Structure

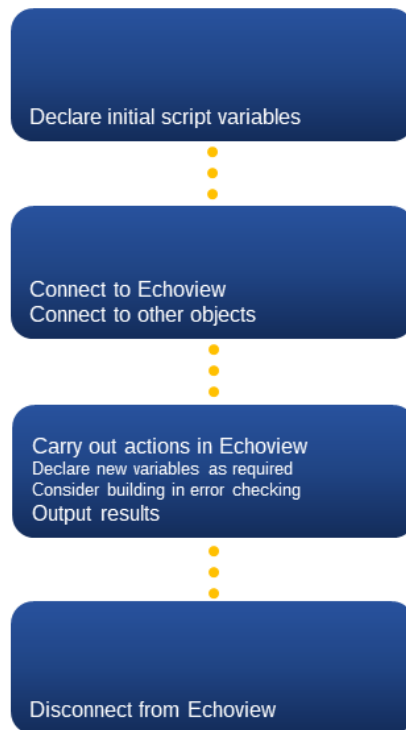


Figure 6. General script structure

A self-contained script has a structure with a beginning, middle and end.

Consider arranging your instructions in modular sections with clear comments. The computer will literally do as you ask. If the conversation between your instructions and the computer fails, troubleshooting or debugging will be made easier by the clear and logical structure of the script.

A variable is a shorthand way of referring to a thing. This is useful when the thing has a long name or is common to some other things.

Consider `CreateObject("EchoviewCom.EvApplication")`. This instruction refers to Echoview. It is tedious to type it out every time you want to use Echoview. Every time you want to do something in Echoview you will need to use it to preface your instruction.

For example, it is used to:

- open Echoview
- write to Echoview's Scripting log file
- get the version number of Echoview.
- close Echoview.

The variable **EvApp** is much easier to use because it can be set to refer to the full scripting object once it is defined in the script and can then be reused repeatedly as a shortcut.

The script `Connect to Echoview.vbs` uses the following Echoview COM interface objects, properties and methods.

Echoview COM interface	Description
.	The full stop is used to join COM interface actions. e.g., <code>EvApp.Version</code> , <code>EchoviewCom.EvApplication</code>
<code>EchoviewCom.EvApplication</code>	This represents Echoview. EchoviewCom is the name of the Echoview COM library. EvApplication is the name of the Echoview Application object.
<code>EvApp.Log</code>	"Log" is a part of EvApplication. It accesses Echoview's script log file.
<code>EvApp.Version</code>	"Version" is a part of EvApplication. It gives the version number of Echoview
<code>EvApp.Quit</code>	"Quit" is a part of EvApplication. It shuts down Echoview.

The script "Connect to Echoview.vbs" uses the following VBA COM interface.

VBA COM interface	Description
<code>CreateObject</code>	A method of the VBA library's Interaction module. It creates an object for or connects to an application.
<code>MsgBox</code>	A method of the VBA library's Interaction module. It displays a Message box with specified text and outputs.
<code>Date</code>	Date is part of the VBA library's DateAndTime module. It returns the date.
<code>Time</code>	Time is part of the VBA library's DateAndTime module. It returns the time.

COM Exec ()

The Automation module allows you to access two interfaces to Echoview. In many cases the Command Interface allows access to more properties than offered by COM object interface. The two interfaces have different levels of support for Echoview COM objects, with limited overlap.

- The COM interface is based on supported Echoview COM objects and the handling of their properties and methods (refer to [Topic 3, section for Overview of Echoview COM objects](#)).
- The Command interface is based on single line commands of string-based syntax. The syntax can identify or change supported Echoview properties, specify supported Echoview actions or filter supported Echoview variable objects.

The COM EvApplication.Exec() method is designed to send commands to the Echoview Command interface.

Example script contrasting COM approaches

```
'-----  
'START SCRIPT  
'-----  
  
Dim EvApp: Set EvApp = GetObject( "EchoviewCom.EvApplication")  
Dim EvFile: Set EvFile = EvApp.OpenFile("C:\Temp\ExampleFile.EV")  
  
'-----  
'EXAMPLE 1: CHANGING A VARIABLE PROPERTIES SETTING  
'-----  
  
'COM approach  
Dim Var: Set Var = EvFile.Variables.FindByName("Fileset 1: Sv raw pings T2")  
Dim VarAc: Set VarAc = Var.AsVariableAcoustic  
VarAc.Properties.Data.ApplyMaximumThreshold = True  
  
'New COM Command Interface approach  
EvApp.Exec ("Sv raw pings T2 | ApplyMaximumDataThreshold =| True")  
  
'-----  
'EXAMPLE 2: CHANGING AN EV FILE PROPERTIES SETTING  
'-----  
  
'COM approach  
EvFile.Properties.Echogram.EchogramMode = 1  
  
'New COM Command Interface approach  
EvApp.Exec ("Ev File | EchogramMode =| Range")  
  
'-----  
'END SCRIPT  
'-----
```

Example caveats

The contrast VBS script uses simplified examples. There is no code to protect against errors that can arise in the execution of the script or handling of the Echoview returns (to script instructions).

Tips:

- Refer to the Help file for Command interface syntax information and error handling approaches.
- Refer to the Echoview website: <https://echoview.com/support/example-scripts/> for a fully developed script – Combine COM and command interface to change variable properties.vbs

What else do you need?

The following tutorial subjects offer further and fuller details on what you need to start:

What do you need to start?	Topic
Knowledge about the Echoview COM interface.	Topic 2: COM Concepts and Terminology Topic 3: Echoview COM Basics Echoview Help file COM object descriptions
Awareness of the process for Echoview tasks.	Topic 5: Create a Script for Echoview
Knowledge about the structure of a script.	Topic 4: Basic VBScript Concepts Topic 5: Creating a Script for Echoview
Knowledge about the syntax and grammar of a script language.	Topic 4: Basic VBScript Concepts Topic 5: Create a Script for Echoview Topic 6: Viewing Echoview COM Objects and Creating Scripts A language's reference information.

Tips:

VBScript language features and added functionality <ul style="list-style-type: none"> • Option Explicit • FileSystemObject • VBA 	<p>VBScript offers features that are peculiar to the language. Some of which are:</p> <p>Option Explicit</p> <p>This term can be used at the very beginning of a script.</p> <pre style="border: 1px solid black; padding: 5px;"> Option explicit ..!Start Echoview Dim EvApp: Set EvApp = CreateObject("EchoviewCom.EvApplication")!Open the Schools Detection EV file Dim EvFile: Set EvFile = EvApp.OpenFile("C:\Myriax\Echoview\Echoview4\Tutorials\COM Scripting\SchoolsDetection\COM_SchoolsDetection.EV") If EvFile Is Nothing Then MsgBox "Failed to open EV file" Else </pre>
--	---

	<p>“Option Explicit” is a special instruction to the VBScript interpreter that goes before all other lines of code. It means that you must explicitly declare all of your variables (i.e., use ‘Dim’). If you do not use Option Explicit, the script will implicitly assume the declaration and allow you to use a variable that you have not declared. In the example above, we have declared the Echoview application to be EvApp. Without the use of Option Explicit, if you later mistype the variable as EbApp, a new variable will be created.</p> <p>It is good practice to put Option Explicit at the very start of your script.</p> <p>Case-insensitive names</p> <p>VBScript method and variable names are case-insensitive. VBScript is not case-sensitive for commands or variable names but case-sensitive for everything else, including strings (text).</p> <p>e.g., MsgBox or msgbox, Option Explicit or option explicit, Dim or dim.</p> <p>FileSystemObject</p> <p>VBScript support includes the Windows FileSystemObject³ model. The FileSystemObject offers a COM interface for files and folders on your computer - files and folders that your script uses.</p> <pre style="border: 1px solid black; padding: 5px;">Dim FSO Set FSO = CreateObject("Scripting.FileSystemObject")</pre> <p>VBA - Visual Basic for Applications</p> <p>This is the language for Microsoft Office applications, and it is embedded in those applications. A *.vbs script can use names (directly) from the language.</p> <p>“CreateObject” is important and is the most prominent - it enables your script to open an application object.</p> <p>For example in Connect to Echoview.vbs, the script uses CreateObject, MsgBox, Date and Time.</p>
<p>Scripting log file</p>	<p>Echoview’s scripting log file can be a useful debugging tool when a script fails because of an error or incorrect output. It records errors and can record messages sent from the script (see the example below).</p> <p>A file called Echoview scripting log.txt is created in the same folder as the Echoview.exe used when you run a script. Typically, the file path will be C:\Users\Public\Documents\Echoview Software\Echoview <Version>\Echoview scripting log.txt</p> <p>The Echoview COM interface enables you to write additional information to the scripting log file.</p> <p>e.g., CreateObject.EvApplication.Log("write this to the log")</p>
<p>Snippets, example scripts</p>	<p>To start writing scripts quickly, take advantage of parts of existing scripts and snippets of code. Examples can be found in the following places:</p> <ul style="list-style-type: none"> • This tutorial • Echoview help file: Echoview COM object descriptions and script examples <p>Echoview website: https://echoview.com/support/example-scripts/</p>

Topic 2: COM Concepts and Terminology

What is COM scripting?

Microsoft COM (Component Object Model) technology is used in the Microsoft Windows-family of operating systems to enable programs to communicate⁴. When you run Echoview, you use the various menu commands to add, remove, process and export data. Scripting allows you to automate these tasks. A script is a series of instructions for an application such as Echoview.

Scripts can:

- save time because they enable you to automate repetitive tasks
- reduce subjectivity and increase the repeatability of results.

Echoview's Automation module allows you to use Echoview COM objects to control Echoview.

Basic concepts and terminology

There are some basic concepts and terminology that you will need to know before you can write a script. The information on this topic is for those new to COM scripting. If you are already familiar with scripting terminology, you can skip to [Topic 3: Echoview COM Basics](#).

To access COM objects through scripts, you use the following COM components:

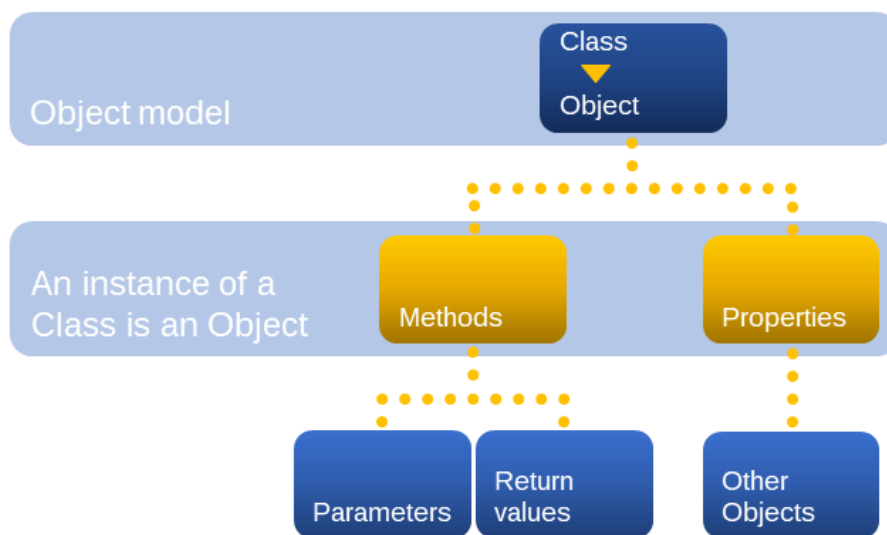


Figure 7. Relationships between COM components.

COM objects and classes

Most COM scripting commands require you to specify a **COM object**. To do that, you will need to know what it is (the name of the object), and where it is located (the **class** to which it belongs).

In programming terms, an **object** is a thing, and an object's **class** is the type of thing it is.

Physical objects and the object's class

Example 1:

Consider a person called John.

- The thing (*object*) is called John.
- The type of thing that John (*class*), is a person.

Example 2:

A salmon is a thing. A particular salmon is tag #1234.

- The object is salmon #1234.
- The class is salmon.

A COM object model

An object model describes how various objects are related to each other within the computer program. The Echoview COM object model describes the relationships Echoview objects have with each other.

Example

Consider a pen. You can hold a specific pen, which is an object. Your pen's class may be Ballpoint. However, your pen consists of several components, which are both objects and classes. The pen has a cap, a barrel and a nib. The specific cap, barrel and nib in your pen are objects which belong to Cap, Barrel and Nib classes.

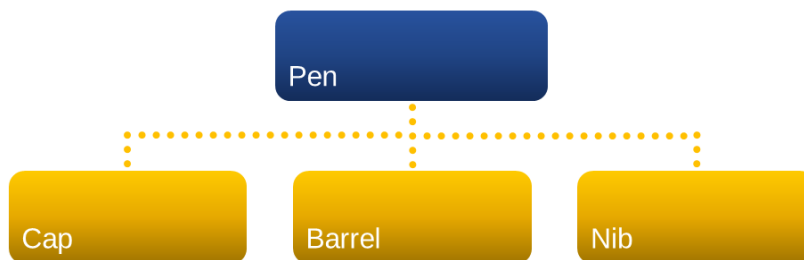


Figure 8. Object model for a Pen

COM properties

A property is a value of an object. Properties can be 'read-only' or 'read-write'.

Example 1: Read-only property

Person objects have a property called mood. They can have values of happy, sad, irritated, maudlin, thoughtful, and ecstatic.

You can ask a person what mood they are in, but not instruct them what mood to be in.

Example 2: Read-write property

If a Person has a property called Shoes, which can have a value of None, Sneakers or Sandals, then you can query which shoes they have on, and instruct them to change to different shoes. This is known as reading the value, and writing or setting the value.

COM methods, parameters and return values

A **method** is an action performed on and by objects.

Person objects can perform many actions. One such action is sleep. If you have full control over the Person class, you can instruct a person to sleep.

Some methods, or actions, require other objects to work. If you instruct a person (object) to eat (method), they will need to eat something – another object.

In some cases, methods will need more than one object. You may want to specify whether they should eat using chopsticks or a spoon. These extra objects that you specify when using a method are called **parameters**.

Methods may give you results, called **return values**. Not all methods will have a return value.

Examples of return values:

- Use the method GiveMeThatRightNowOrElse on a child object, and it will return an object or nothing (if the child does not have anything).
- Use the method CountToes on a person object and it will usually return the number 10.
- Use the method AscendMountainSummit on a person object, and it will return either success or failure.

Topic 2 exercises

To reinforce your understanding, the following exercises pose key questions and are accompanied by reasoned answers.

For the following example, write which parts of the words/phrases fall into the categories for object, class, method, parameter, and return value. Instruct Tim from the work gang to climb to the platform in the oak tree using the rope ladder, and bring down my binoculars. Ensure that he is wearing his climbing shoes and report to me when he is done.

Object	Tim, platform, oak tree, binoculars
Class	Work gang
Method	Climb, wear
Parameter	Rope ladder
Property	Climbing shoes
Return value	Report to me

1. What does a class describe or say about an object? What is the difference between an object and a class, and why is it important to understand their relationship?

The class of an object specifies the type of object it is, e.g., a tree. The phrase “is-a” is used when referring to objects and classes. For example, ash tree (object) is a tree (class).

The class hierarchy provides you with increasing detail about an object – for example, that is an ash tree as opposed to just a tree.

In relation to Echoview, it might specify that a variable is a virtual variable rather than a raw variable, for example.

Once you have identified an object's class, you then know what can be done with an object. There are certain actions that can be performed on virtual variables that cannot be applied to raw variables. The methods and properties of virtual variable objects are different to those of raw variable objects.

2. Can an object have more than one class?

Yes, an object can have more than one class. The class describes what the object is – it may be two things, or it may be a specific version of a thing. For example, an acoustic virtual variable is both an acoustic variable and a virtual variable. This is known as class inheritance, or class hierarchy – you may start with the variable class, and then work down to more specific types of variables that only apply to some. Each of these types is a sub-class.

3. To determine the number of toes a person has, which would be better – method CountToes or property NumberOfToes?

A method generally performs an action. The method CountToes would give the value for a particular object, John. As an individual, John may have greater or less than 10 toes.

A property is an attribute of an object. The property NumberOfToes would give the expected value (property) for a particular object. The property value would be 10 – the expected value for the number of toes for a Person object.

If you needed to count the number of toes each time you queried the dataset, you would use a method. However, you can assume that a specific person will always have the same number of toes, and they do not need to count their toes every time you ask them. You are simply querying the value, not performing an action.

While both a method and a property would achieve the same thing (telling you the number of toes a specific person has) a property is more appropriate.

4. Can an object have a property that is another object?

Yes, properties are information about an object. Consider a Person object – it could have a Head property, and that head is in itself an object, which belongs to the Person.

5. Can a method that is set to return something, return a value of nothing? What is the difference between this and a method that has no return value?

Yes, a method can return nothing.

Some methods do not have a return value, they never return anything.

There are other methods that have a return value (such as the result of searching for a variable with a particular name) – it returns the variable object. If these methods do not return anything, they return a special “null” value that represents “nothing”.

Further reading

Objects, classes and methods are explained in Microsoft's *Sesame Script* series⁵.

Topic 3: Echoview COM Basics

The discussions in this topic deal with the relationships of Echoview COM Objects to the actual features of the Echoview program, and how to interact with Echoview through a script.

Register Echoview COM

When Echoview COM objects are requested (by scripts), Microsoft Windows finds the COM object in the registered version of Echoview. When there is a mismatch between Echoview COM and the Echoview version, incorrect script error messages may be displayed when running (known error-free) scripts.

To find out which version of Echoview is registered:

- On the **File** menu, select **Echoview Configuration**.
- Click on the **Scripting page** to view your version details.

Note: After installing it you must run Echoview once using **Run as Administrator** to register COM scripting. You will need to do this for each version or edition of Echoview that you install.

The Echoview Component Object Model (COM)

The Echoview COM is an interface to access settings and actions in Echoview. The Echoview team creates COM classes to access selected parts of Echoview. Echoview COM support evolves and grows with each Echoview version.

The Echoview COM has a hierarchical structure and shows you the relationships between, and path to, each COM class. Below is a simplified version of part of the Echoview object model.

Echoview may have one or several files open. Each EV file can have one or more (raw or virtual) variables. Each variable can have one or more pings.

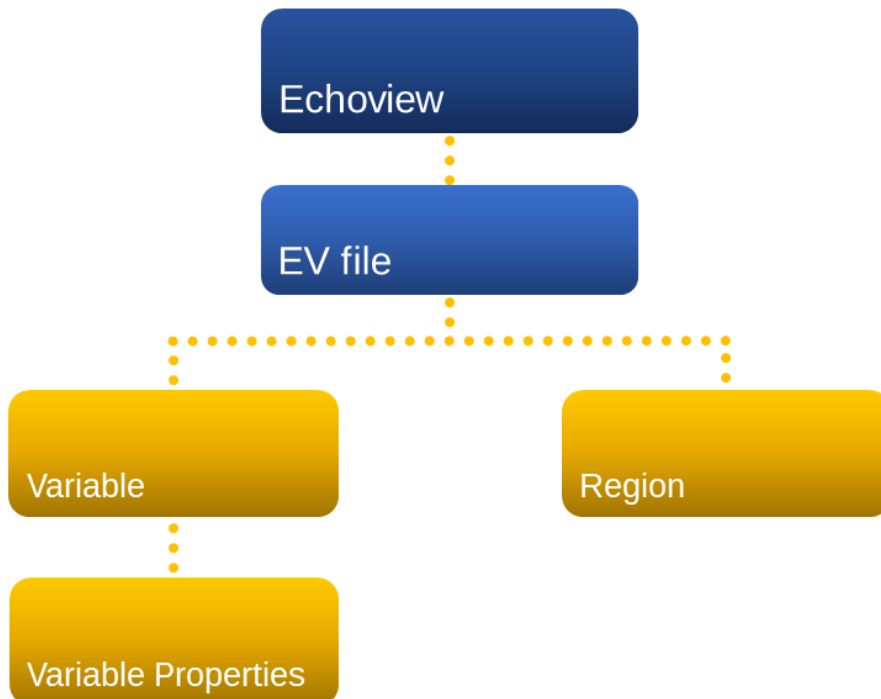


Figure 9. Partial and simplified Echoview COM object model in terms of Echoview entities.

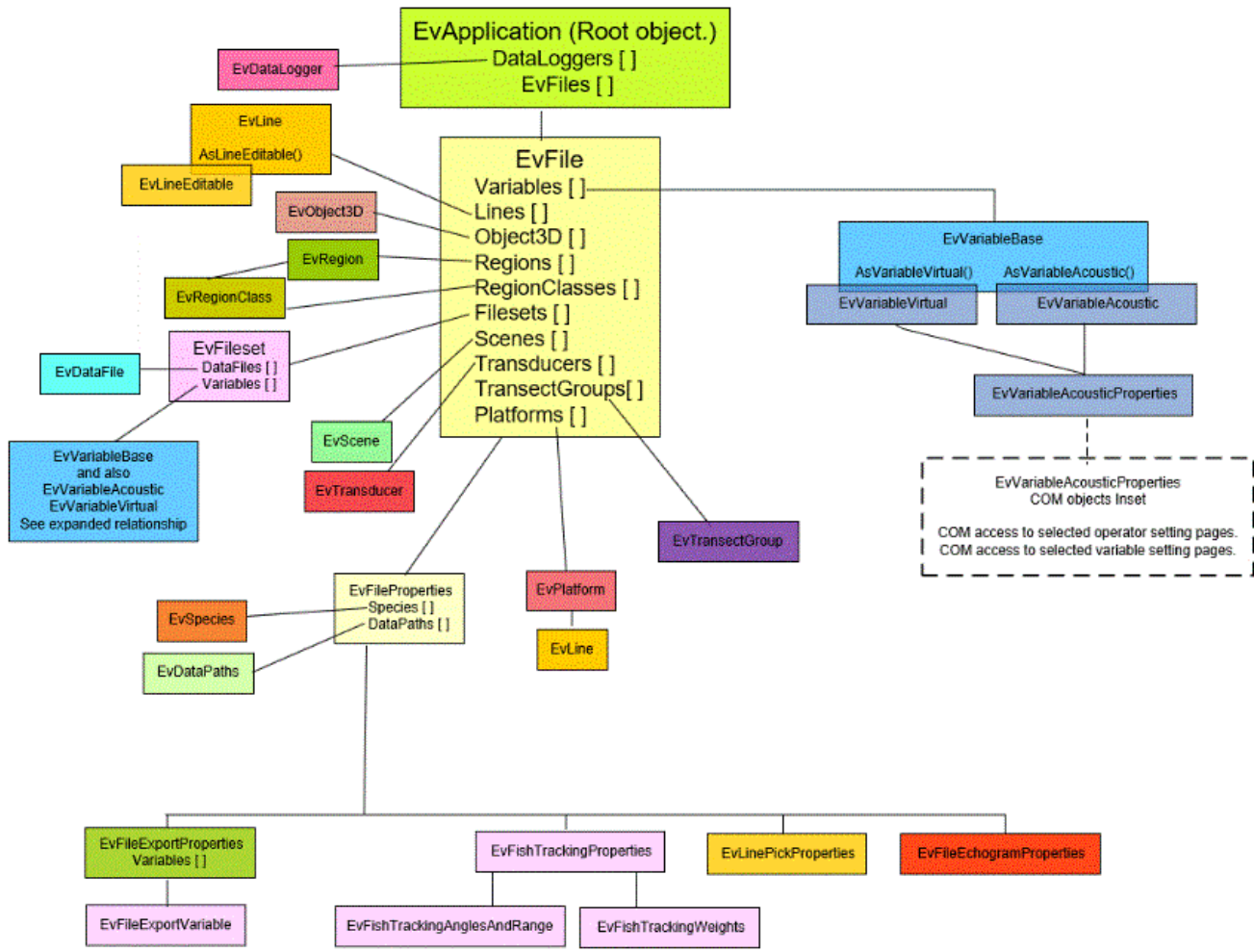


Figure 10. A diagram from the COM object hierarchy page in the Echoview Help file. Displayed are relationships between the COM classes (boxes) and collections (names with []). Each COM class and collection has its own methods and properties (not shown).

Echoview COM objects are accessed through the COM class hierarchy. Information about an Echoview version's COM hierarchy, COM Objects and their Methods and Properties can be found in the Echoview help file (information applicable to a specific version of Echoview) or Echoview web help (information applicable to the latest version of Echoview).

The Echoview COM is very large and complex. In the following examples, we will only look at a small subset of the model.

Reminder: a COM object is a specific instance of a COM class. For example:

Echoview COM class	Echoview COM object
EvApplication	The open Echoview program.
EvFile	AutoOpen.EV
EvVariable	Sv pings
EvDataFile	"C:\Users\Public\Documents\Echoview Software\Echoview 14.0\Example data\96082826.ELD"

The EvApplication class

EvApplication represents the main program. You may think of it as the main Echoview window, but in reality, it is the underlying program.

EvApplication performs few actions. It has a [property](#), which is the version number. One of its other functions is to open an EV file. You can only have one EvApplication Object in a script. This is because you can only control one instance of Echoview. The first lines of your script will be the lines to get this COM object.

Access to other Echoview objects

To access an Echoview variable (raw or virtual) or Echoview setting, you start by referring to the EvApplication Object, then the EvFile object.

Supported Echoview variables and variable property settings are accessed from within the EvFile object (using the variables property), the EvVariableBase, and the EvVariableAcoustic and/or EvVariableVirtual objects.

Echoview EV file features (Lines, Regions, Filesets) and settings (EV file properties) are accessed from within the EvFile object.

Echoview classes

Echoview's COM objects and classes are prefixed with Ev. For example:

- **EvApplication** is the class for the main application.
- **EvFile** represents an EV file.
- **EvRegion** represents a region.

Tips:

To create a script, work down the object model, starting with the main object (the application). Use the Echoview COM object hierarchy from the Echoview help file to identify the objects you need to access.

Echoview objects

An Echoview COM Object is an instance of an Echoview COM Class.

Echoview COM Objects refer to specific files, things or settings within the program, for example:

- AutoOpen.EV
- Region A
- A line pick called BestBottomLine
- A line at a fixed depth of 5m
- Files in Fileset 1
- Sv raw pings T1

Echoview object properties

The Echoview help file details the Properties for each Object. For example, some of the object properties for EvApplication (the Echoview application) are listed as follows:

EvApplication properties	Description
EvFiles	EvFiles [] <i>Summary</i> Read-only EvFilesCollection object containing a reference to an EV file for each open document.
Version	Version <i>Summary</i> A string representation of Echoview's version number. Read-only

The `EvFiles[]` property gives access to the Echoview file collection (described under [Echoview collections](#)) i.e., the EV files that are open on your computer.

The Version property is Echoview's version number, expressed as a string. It is a read-only value of [simple type](#). The version number has three decimal points that separate parts of the version information e.g., 4.70.39.1327. The version is interpreted as a string (not a number).

Echoview object methods

Echoview object [methods](#) support actions that the program performs, e.g., Open an EV file, run Detect Schools.

One of the Methods for the EvApplication Class is OpenFile. What this method does is open an EV file.

EvApplication Method	Description
OpenFile	EvFile OpenFile(string FileName) <i>Summary</i> Opens an EV file and returns the EvFile object for it. <i>Parameters</i> FileName The path to an existing file to open. <i>Return</i> A COM object referring to the newly opened document.

The OpenFile method has one parameter, a string. In this example, the parameter is called FileName. The convention is to specify the type of parameter that is expected, followed by a descriptive name. If there is more than one parameter, the parameters are separated by commas. An example for setting more than one parameter is in **Example 5.2**.

FileName specifies the path to the EV file you want to open:

E.g., `C:\Echoview Software\Tutorials\GettingStarted\GettingStarted_Topic1.EV`

The [return value](#) is an EvFile object, which is the EV file that was opened, but it could also be nothing (null) if it couldn't open the EV file.

More useful methods are:

EvApplication methods	Description
NewFile	<p>EvFile NewFile(string Filename)</p> <p><i>Summary</i></p> <p>Creates a new EV file and returns the EvFile Object for it.</p> <p><i>Parameters</i></p> <p>Filename.</p> <p>An optional parameter, used for specifying a template file.</p> <p><i>Return</i></p> <p>A COM object referring to the newly created document.</p> <p>Note: If a template file is specified, this method will first check for a user-specified templates folder path. If unsuccessful the method will check the default path for the templates folder.</p>
CloseFile	<p>CloseFile(EvFile File)</p> <p><i>Summary</i></p> <p>Closes an open EV file. Closes the file connected to the EvFile object.</p> <p><i>Parameters</i></p> <p>The EvFile object that refers to the EV file to be closed.</p> <p><i>Return</i></p> <p>None.</p>
Exec	<p>string Exec(string CommandLineText)</p> <p><i>Summary</i></p> <p>Sends the string for a single command line to the Echoview Command interface. The return for Exec is a string sent from the Command interface. A successful command line may return text output or an unsuccessful command line may return an error message. In either case, to interpret the return string from the Command interface use Exec return handling functions. See also Topic 1: COM Exec ()</p> <p><i>Parameters</i></p> <p>CommandLineText</p> <p>A text string that is a single command line written in Command interface syntax for the Echoview Command interface.</p> <p><i>Return</i></p> <p>A text string returned from the Echoview Command interface.</p>

Echoview collections

A Collection is a special type of object used to hold other objects and provide access to these other objects.

For example: a fileset has a number of data files AND a number of raw variables.

Echoview COM uses collections to hold similar objects. Each collection holds a different set of objects (such as EV files, raw variables), but the collections are all accessed in the same way.

You can iterate through a collection by accessing items 0 to count -1 . A collection's index starts at zero. An index is a number that you use to indicate the n^{th} item in a collection.

For example:

- Filesets(0) is the first fileset.
- Filesets(1) is the second fileset.
- Filesets(Count -1) is the last item in the fileset, where Count is the number of items in the collection.

For all Echoview collections, you can:

- ask for a number of items
- get a specific item by its number (e.g., 'give me variable no. 5')
- find an item by its name (e.g., 'give me Sv raw pings T1').

Some Collections will allow you to perform additional actions, depending on the Objects they contain.

All Collections have a FindByName method that uses the name of the object as a string. This makes it simple to find an object if you know what it is called. Most objects have this Name property.

The collection object is a Property. The EvApplication class has a property called EvFiles, which is a collection object that lets you access EV files.

In the Help file, collections are written with two square brackets after their name e.g., EvFiles []. The brackets indicate that the object holds a number of objects.

Overview of Echoview COM objects

Not all of the functionality in Echoview may be accessible via COM scripting. Accessing the features or settings of Echoview using COM requires explicit code support. The latest Help file details all current COM support and latest additions. If there is COM functionality that you would like to see added to Echoview, please contact [Echoview Support](#).

To get basic information about Echoview's COM objects, you can use an object browser, associated with a text editor program. See [Topic 6: Navigating Echoview COM Objects](#).

Help file topic page name	Description
Echoview COM object hierarchy	Hyperlinked schematic of the supported Echoview COM classes.
Summary of COM objects	Hyperlinked list of COM classes with descriptions of the Echoview feature or settings they support.
COM object and COM object collection pages	Every COM object has its own page listing methods and properties.

Echoview variables

An EvFile object contains a variables collection that includes all the raw and virtual variables in an EV file. It is equivalent to all the variables shown on the **All** tab of the Dataflow window.

An EvFileset object also contains a variables collection, comprising the subset of variables in that fileset (the ones that are shown in the Fileset window).

EvVariableBase is the base class that contains all variables, and its methods apply to all of the classes it contains. In effect, EvVariableBase gives you access to variables within EvVariableAcoustic and EvVariableVirtual.

EvVariableBase allows access to an Echoview variable to:

- determine variable type (e.g., acoustic, movement, position, and heading)

- change the variable's name and short name
- determine whether or not the variable is usable
- export variable data to a CSV or other file format.

Access to the variables allows you to modify settings that you may have done via a template in the past, including:

- Iterate (go through) all the variables in a file (or fileset) and find the ones you specify using any criteria.
- See a list of variables in a familiar way (as a fileset or list of all variables).
- Change common variable settings such as data thresholds and grids.
- Automatically change the settings of variables in a virtual variable chain and import EV files.
- Perform any export available in the user interface.

Acoustic variables

An `EvVariableAcoustic` object gives access to a variable's variable properties, and supports all creation and export methods, including:

- Creation of line-relative regions, 3D fish tracks and single targets.
- Detection of fish tracks, schools and surfaces.
- Pick lines.
- Exporting lines to CSV.
- Exporting data to the Echoview data file format and HAC.
- Exporting underlying data, integrations, region data, and frequency distributions.
- Finding the position variable for the platform.
- Finding the data type, and number of pings of a variable.
- Using the Properties property to change settings on the grid, data, display, analysis, and calibration pages of the variable properties.

Properties also allow you to access some of the virtual variable operator settings.

Regions and region classes

COM scripting gives you access to all regions and region classes in Echoview. Many methods require you to pass in a region or region class object.

Use `EvFile.Regions []` to:

- import and delete regions
- access a specific region
- find a region by name.

Use the `EvRegion` object to:

- export that region
- change its type, name and/or notes
- read its start and stop times
- determine whether it is 2D (single beam) or 3D (multibeam).

Note: You cannot edit a region's shape using COM.

Use `EvFile.RegionClasses []` to:

- find a class by name.
- add and delete classes.

Use the `EvRegionClass` object to:

- export all regions of that class.
- delete all regions of that class.
- delete a specific type of region in that class.
- change the name of the class.

Lines

Use `EvFile.Lines []` to access all lines in an EV file to:

- change a line's name.
- export to an EVL or CSV file.
- determine whether an object is a line (`AsLineBase`).
- create and delete editable lines.
- determine whether a line is editable (`AsLineEditable`).
- overwrite an editable line (`OverwriteWith`).

Filesets and data files

Use COM scripting to automate adding filesets to an EV file and adding datasets to that file.

Use `EVFile.Filesets []` to access filesets to:

- find a fileset by name (`FindByName`), which allows access to a fileset's `DataFiles []`.
- read and set a fileset's time offset.

Use `DataFiles []` to:

- access a fileset's data files and variables.
- add and remove data files in a fileset.
- access a specific `EvDataFile` object that, in turn, gives access to the file's filename and first and last ping times.

EV file properties

Use the `EvFileProperties` object to access:

- 2D schools-detection settings.
- fish-tracking settings.
- export settings on the Export page of the EV File Properties dialog box (including export variables, export mode, and whether to export empty pings).
- data paths where Echoview searches for data files.
- species settings.
- line-pick settings.

Examples of use include:

- changing fish-track or schools-detection settings before performing a detection – by doing this you can perform several detections with different settings in succession and compare the results.
- changing which export variables are included in exports, and which format they are exported in.
- changing the file path for where Echoview looks for data files (a useful method if you move your data files).

From Echoview COM class descriptions to objects in a script

Main task: Export the integration results from a region by cells analysis for a single region from an acoustic variable in Echoview.

```

Option Explicit
Dim vEchoviewCom: Set vEchoviewCom = CreateObject("EchoviewCom.EvApplication")

Dim vEvFile: Set vEvFile = vEchoviewCom.OpenFile("C:\Temp\biomass.ev")

Dim vEvVariablesCollection: Set vEvVariablesCollection = vEvFile.Variables
Dim vEvVariable: Set vEvVariable = vEvVariablesCollection.FindByName("sv split beam pings (channel 3)")

Dim vEvRegionsCollection: Set vEvRegionsCollection = vEvFile.Regions
Dim vEvRegion: Set vEvRegion = vEvRegionsCollection.FindByName("Region2")

vEvVariable.ExportIntegrationBySingleRegionByCells "C:\Temp\Output\integration region.CSV", vEvRegion
  
```

The code snippet is annotated with several callouts:

- Ignore undeclared script variables (typos)**: Points to the `Option Explicit` line.
- Open Echoview**: Points to the `CreateObject` line.
- Which EV file?**: Points to the `OpenFile` method call.
- Which acoustic variable in this EV file?**: Points to the `FindByName` call for `vEvVariable`.
- What region do I select with the mouse?**: Points to the `FindByName` call for `vEvRegion`.
- Analysis export by single region by cells for this output file and this region on this variable of this EV file.**: Points to the final `ExportIntegrationBySingleRegionByCells` call.

Figure 11. Annotated snippet of code.

Script variables are used as shorthand labels for specific COM objects. Red text is for the first occurrence of the variable and blue text is for each subsequent use of a variable that was previously defined.

`vEvVariable` is effectively:

```

CreateObject("EchoviewCom.EvApplication").OpenFile("C:\Temp\biomass.ev").
Variables.FindByName("sv split beam pings (channel 3)")
  
```

In other words:

The Sv split-beam pings (channel 3) variable in the `C:\Temp\biomass.ev` EV file in Echoview.

The relationship of the COM classes for the code snippet (above):

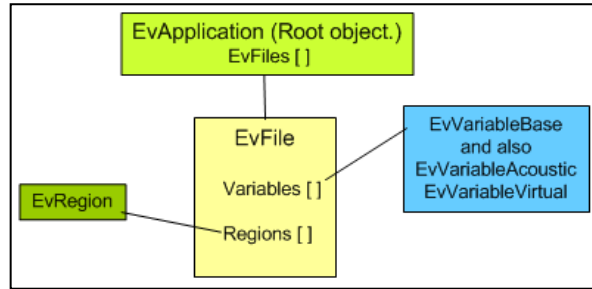


Figure 12. COM class relationships for the code snippet.

Topic 3 exercises

To reinforce your understanding, the following exercises pose key questions and are accompanied by reasoned answers.

1. How many Echoview application objects can you have in your script?

You can only access the Echoview application once within a script.

Note: Be careful when using Echoview at the same time a script is running. This can cause problems when the user usage of Echoview clashes with the script's actions (especially with opening and closing EV files and changes in dialog box settings). Usually, the script will crash and give an error in this situation, but Echoview should never crash no matter what you do.

If you wish to use Echoview while a script is running, you should open a second copy of Echoview once the script has started.

2. Using the search function in the Echoview Help file, fill in the missing parts of the following table.

Hint: you can also refer to Topic 5 in this tutorial and use the Microsoft Visual Basic browser to examine Echoview COM interface.

3.

Method: EvLine.Export	Parameters ...
Object: EvFile	Collections ... Properties and methods ... Classes ...
Object: EvCalibration	Methods ...
Method: DetectSchools	Object: EvVariableAcoustic or EvVariableVirtual Parameters: ...

4. What is the class for EvApplication?

EvApplication.

5. How many variables can you have referring to your EvApplication object/s?

Any number of script variables can refer to the same Echoview application object. But only one EvApplication object is allowed in a script.

6. How many variables can refer to the EvFile Object?

Any number of script variables can refer to the same EvFile object.

7. What is a collection, and why does Echoview have collection objects?

A collection is an object that contains, and provides access to, a set of objects.

For example: The COM class EvDataFilesCollection gives you information about and access for adding and removing data files (as data file objects) in an EV file.

The EvVariablesCollection class gives you access to the set of raw variables (as variable objects) in an EV file.

8. Can a collection object be a property of an object?

A collection object is a property of an object. Using the collection examples from the previous question:

The EvFileset object has a property called datafiles. If you look it up in the help file, you will see that it is followed by two square brackets 'Datafiles []', which indicates that it is a collection of objects. Clicking the link will take you to the EvDataFilesCollection.

The EvFile object has a property called variables. If you look it up in the help file, you will see that it is followed by two square brackets 'Variables []', which indicates that it is a collection of objects. Clicking the link will take you to the EvVariablesCollection.

9. Why would it be useful to have a return value for the method 'DetectSchools'? What would you want the method to return?

A return value is used to obtain information from a method.

If you do a schools detection, you may want to know how many schools were detected or whether any schools were detected.

Topic 4: Basic VBScript Concepts

You can use scripts to access COM objects with a number of programming languages. In this tutorial, we will use VBScript, which is the language used by .vbs files.

VBScript syntax

Computer languages have their own grammar or syntax. This means that your instructions must be phrased in a particular way for the computer to understand them. Computer languages are very precise, and if you do not construct your instructions according to the syntax rules, you will get a syntax error. This means that you have phrased something incorrectly.

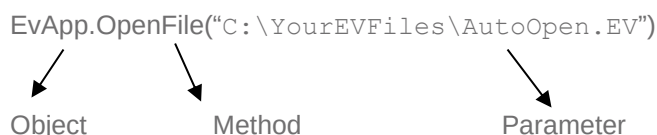
Accessing properties and methods

To access an object's property, you type the object followed by a period and then the method or property. A script variable that refers to a COM object can also use the period to access further COM objects, properties or methods.

For example, to access the version number of your Echoview application, you would query the version property of the EvApplication object. The EvApplication object is referred to as the script variable EvApp (which is defined earlier in the script): EvApp.Version

To access an object's method, you follow the same rules, but also add brackets around the parameters. The example below uses the method OpenFile to open AutoOpen.EV:

```
EvApp.OpenFile("C:\YourEVFiles\AutoOpen.EV")
```



The diagram shows the code `EvApp.OpenFile("C:\YourEVFiles\AutoOpen.EV")` with three arrows pointing to labels below: `EvApp` points to "Object", `.OpenFile` points to "Method", and `("C:\YourEVFiles\AutoOpen.EV")` points to "Parameter".

Brackets and parentheses

Brackets/parentheses around parameters aren't always necessary - it depends on the situation. Bracket usage may also depend on the scripting language. For example: VBS can be particular about when you use brackets for functions ⁱ and when you don't.

For more information, refer to the Microsoft VBScript Language References^{6,7}:

Using comments

You can write comments in scripts. Comment-text are lines of text that the script ignores. Comments are useful in that they explain what your script is doing. To create a comment using VBScript, put a single quote before the first word.

```
'This is a comment.
```

```
'Start Echoview and open the krill processing file.
```

Note: Always type single and double quotes using the keys adjacent to the ENTER key on your keyboard. If you copy and paste from another program, they may look the same but not translate properly, and the script will not recognize them.

Variables and keywords

Keywords

A key part of the syntax of a language is its **keywords**. Such words have specific meanings in a language. A script is constructed from these keywords and your instructions.

Script variables

Important: The use of the term variable in relation to COM scripting is different from an Echoview variable.

A variable is a way of referring to an object. It is analogous to a person's name.

Example – variables in a list:

Mary
John
Fred

In this analogy, you may refer to a person called Fred. You do so with the concept 'Fred'. A variable is not an object's name. Using a variable is a way of referring to the object. In this case, Fred is the person's name, but it is also a way your mind has of holding a concept of a specific person. When thinking about variables and objects, Fred (the person's name) is a property of the person (object).

Another way of referring to the person would be: 'the third person on the list'.

You can also use a symbol to express a variable, such as 'x' or 'Fred' – this symbol stands for a value.

In most computer languages, when you want to refer to something, you have to do it in two steps:

10. State that you want to refer to something, i.e., say you will have a variable and that it will be called something. This is known as 'declaring a variable'. And the line of code that does this is the 'variable declaration'. You only need to declare each variable once within a script.
11. State what the variable will refer to, i.e., say 'this variable, that I typed as "Fred", means that person over there'. This is known as:
 - 'initializing' the variable the first time you do it in the script
 - 'setting' the variable for subsequent occurrences within the script

You can set a variable any number of times – 'this variable, that I typed as "Fred", is that person over there on the right'; 'now this variable, that I typed as "Fred", is that other person to the left'.

Example – math variables:

Another analogy can be found in algebra. Take an algebraic equation:

$$y = x^2 + 3$$

Using the specific value:

$$\text{Let } x = 5$$

Using the value for x, you can determine the value of y. In this case, y and x are variables, which are implicitly declared. In computer language, the line starting 'Let ...' says that the variable called x has a value of 5. 'Let' is a keyword in this example. When you read it, you know that there is a variable x, and that it will be initialized to some value.

Using variables in a script

There are two basic types of variables in VBScript:

1. Object types (objects with classes).
2. Simple types (numbers and text).

In VBScript, you use two keywords, Dim and Set, for the two stages of declaring and setting a variable.

Dim declares or, in VB terms, dimensions a variable.

Set sets or initializes a variable's value.

Declaring an object type variable

To go back to the algebraic example, for an **object type**:

```
Dim x  
Set x = Fred
```


You can now say *x*, where before you would have said Fred. Fred is the COM object known to Microsoft Windows on your computer. In a script, wherever you use *x* in an instruction, Windows will use to the object Fred.

The `EvApplication` object represents the Echoview application. To declare it, you could use `EvApp` –

```
Dim EvApp
Set EvApp = CreateObject("EchoviewCom.EvApplication")
```

Using Option Explicit

Option Explicit is a special instruction to the VBScript interpreter, which goes before all other lines of code. It means that you must explicitly declare all of your variables (i.e., use `Dim`). If you do not use `Option Explicit`, the script will implicitly assume the declaration and allow you to use a variable that you have not declared. In the example above, we have declared the Echoview application to be `EvApp`. Without the use of `Option Explicit`, if you later mistype the variable as `EbApp`, a new variable will be created.

It is good practice to put **Option Explicit** at the very start of your script.

Declaring a simple type variable

Simple types are things such as numbers and text. VBScript interprets numbers simply as numbers. They are not objects, and they do not have classes, methods or properties. Variables that represent a simple type are declared with a slightly different syntax. You need to declare the variable, but you do not use `Set`.

To declare and initialize a variable called *x* with a value of 5:

```
Dim x
x = 5
```

Strings, which represent text, are also simple types. A string is enclosed by quotation marks:

```
Dim x
x = "Hello"
```

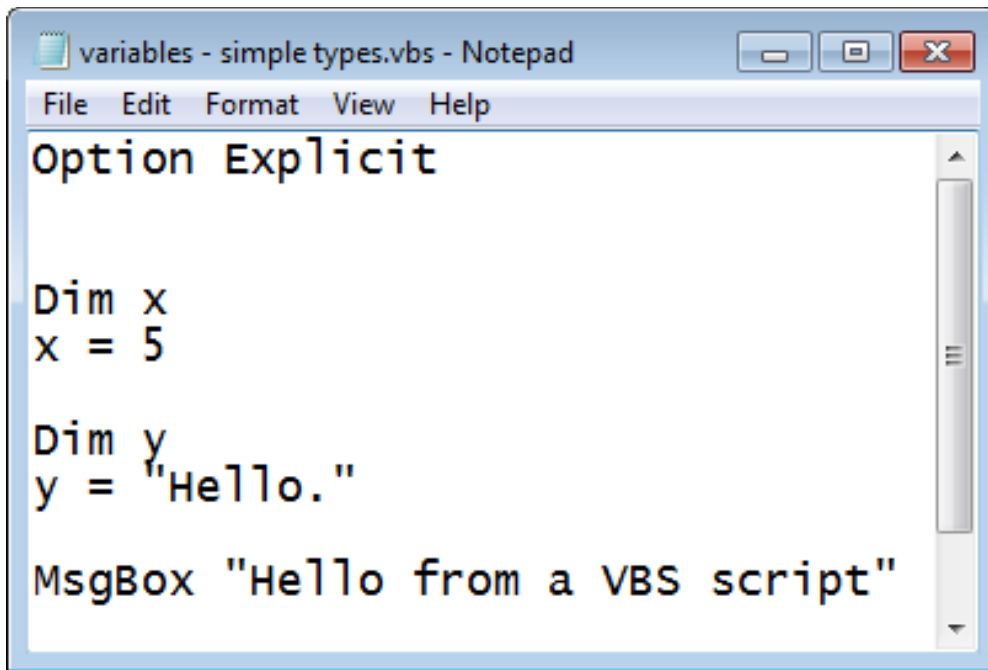
Example 1:

Run the script `variables – simple types.vbs` by double-clicking on the file:

```
C:\Echoview Software\Tutorials\IntroToComScripting\variables - simple types.vbs
```

Result:

A message appears: **Hello from a VBS script**. This is a successful result, as the script is required to send a message to the screen. You can open the script in Notepad to view it. You will see that it declares two simple type variables:



```

Option Explicit

Dim x
x = 5

Dim y
y = "Hello."

MsgBox "Hello from a VBS script"

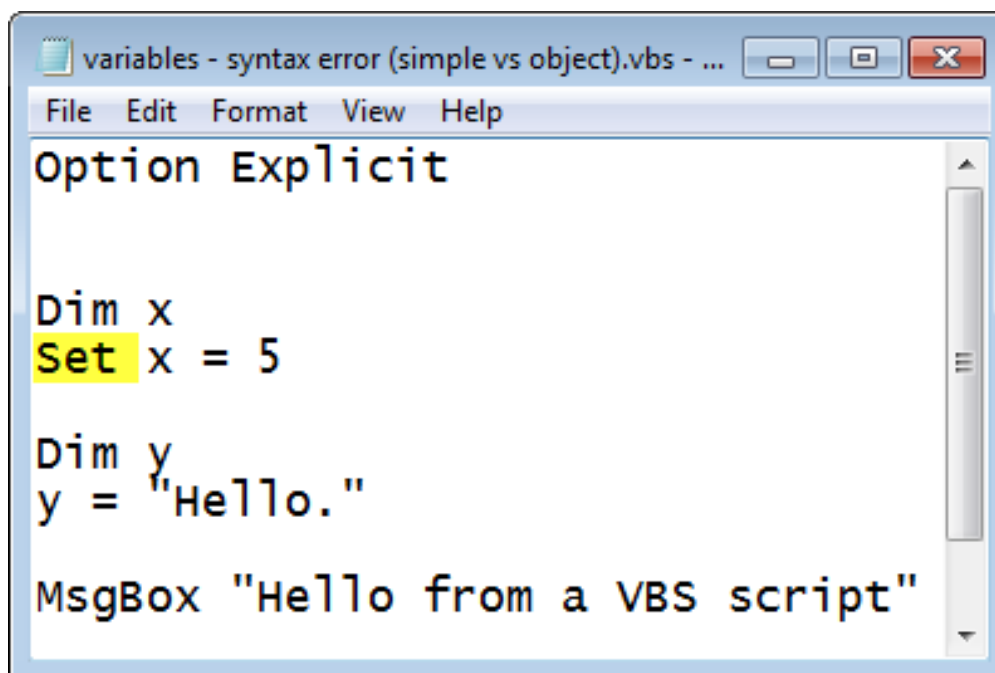
```

Figure 13. VBScript 'Simple Types'

Example 2:

Run the script variable – syntax error (simple vs object).vbs by double-clicking on the file:

C:\Echoview Software\Tutorials\IntroToComScripting\variables - syntax error (simple vs object).vbs



```

Option Explicit

Dim x
Set x = 5

Dim y
y = "Hello."

MsgBox "Hello from a VBS script"

```

Figure 14. The script for "variables - syntax error (simple vs object).vbs".

Result:

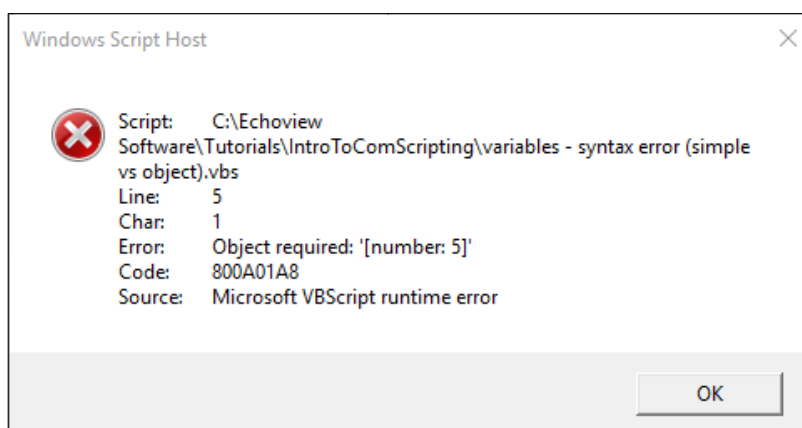


Figure 15. Script error message.

The dialog box gives you detailed information about the syntax error.

Highlighted message	Description
C:\Echoview Software\Tutorials\IntroToComScripting\variables - syntax error (simple vs object).vbs	The file name of the script you are running
5	The line number of the syntax error.
1	The character number of the syntax error. In this case, character 1 is the beginning of the line for the 'Set' statement
Object required: '[number: 5]'	A message which means that the computer was expecting an object, but instead it got a number 5.

The script that produced the syntax error is displayed in Figure 14.

The script contains a deliberate syntax error. The code is written to initialize a variable that is going to be an object (using Set), but the script then sets that variable to a simple value (with x = 5). To set the variable to an object, use Set x = Fred, given that Fred is an object recognized by the computer.

Scripting procedures

Scripts run by the Windows Scripting Host (VBS scripts) and scripts using the VBScript language have additional features you can make use of. For example, there is a MsgBox method that you can call to show a message box that displays the results of COM tests/outcomes for debugging. This method doesn't belong to any class, but simply exists. We call this kind of method a **function** (if they return a value, as MsgBox does, it returns whichever button the user clicked), or a **procedure** (if they don't have a return value).

Tip: Use MsgBox "Reached here" progressively and at any point in a script, in order to determine the line(s) where a script fails.

Sub procedures (AKA procedures)

Sub procedures can take arguments and perform actions, but they cannot return values. A sub procedure starts with Sub and ends with End Sub and encloses a series of statements. To exit a procedure early, use Exit Sub. The sub statement is always followed by parentheses, which contain the arguments. If there are no arguments, these are left empty.

Example of a sub procedure:

```
'A subroutine to export from the named file
Sub ExportFromFile(FileName)
    Dim EvFile: Set EvFile = EvApp.OpenFile(FileName)
    If EvFile Is Nothing Then
        MsgBox "Failed to open EV file '" & FileName & "'"
        Exit Sub
    End If
    ExportVariable EvFile, "Sv Q1 telegrams T1"
    ExportVariable EvFile, "Sv Q2 telegrams T2"
    EvApp.CloseFile(EvFile)
End Sub
```

Other scripts that use subroutines include:

- C:\Echoview Software\Tutorials\IntroToCOMScripting\A_single_export.vbs
- EV script 003 - LINES - PICK and EXPORT in all EV files in a folder.vbs from [Echoview's example scripts](#).

Function

A function can perform actions, return values and take arguments that are passed by calling a function. The mathematical analog is $f(x) = \text{value}$. A function starts with Function and ends with End Function and encloses a series of statements. The function statement is always followed by parentheses, which contain the arguments passed to the function. If there is no argument, these are left empty.

Example of a function:

```
'A function to create the output file name
Function CreateFileName(FileName, VarName, OpName)
    Dim FolderName: FolderName = "C:\Echoview
Software\Tutorials\IntroToComScripting\Script1Output\"
    Dim File: Set File = fso.GetFile(FileName)
    CreateFileName = FolderName & Left(FileName, Len(FileName) - 3) &
    "-" & Left(VarName, 2) & Right(VarName, 1) & "-" & OpName & ".csv"
End Function
```

A value is returned using the function name, i.e., CreateFileName = FolderName...

For further information on Functions refer to the VBScript User's Guide⁸ and Herong's Tutorial Examples⁹.

Note: **fso** is the VBScript file system object that accesses the Windows file system using the Microsoft Script Runtime's FileSystemObject model¹⁰.

Conditional statements

There are four conditional statements used in VBScript to perform conditional actions:

1. **if** Runs a set of code when a condition is True.
2. **if ... then ... else** Selects one of two sets of lines and runs it.
3. **if ... then ... elseif** Selects one of many lines and runs it.
4. **select case** Selects one of many lines and runs it.

Example of use:

The script is a Visual Basic Script file: Echoview Software\Tutorials\IntroToComScripting\A_single_export.vbs:

The script:

- Opens Echoview.
- Opens COMScripting_Topic4.EV and includes a check if the EV file failed to open.
- Opens an acoustic variable called “Sv pings” and includes a check if this variable is opened and if it is an acoustic variable.
- Outputs an Analysis by Regions export to a file FileA-Sv1-ARIN.csv, and checks if the export failed.

```
.....  
' A single export  
.....  
' Open the EV file  
Dim EvApp: Set EvApp = CreateObject("EchoviewCom.EvApplication")  
Dim EvFile: Set EvFile = EvApp.OpenFile("C:\Myriax\Echoview\Echoview5\Tutorials\COM Scripting\COMScripting_Topic4.EV")  
If EvFile Is Nothing Then  
    MsgBox "Failed to open EV file"  
Else  
    .....  
    ' Get the variable  
    Dim Var: Set Var = EvFile.Variables.FindByName("Sv pings")  
    If Var Is Nothing Then  
        MsgBox "Failed to find variable"  
    Else  
        Dim VarAc: Set VarAc = Var.AsVariableAcoustic  
        If VarAc Is Nothing Then  
            MsgBox "Variable is not acoustic"  
        Else  
            .....  
            ' Export the variable  
            If Not VarAc.ExportIntegrationByRegionsAll("C:\Myriax\Echoview\Echoview5\Tutorials\COM Scripting\SingleExportOutput\FileA-Sv1-ARIN.csv") Then  
                MsgBox "Failed to export by regions"  
            End If  
        End If  
    End If  
End If  
.....  
' Tidy up  
EvApp.CloseFile (EvFile)  
EvApp.Quit
```

Figure 16. Snippet displayed in a window under the Microsoft Visual Basic editor, text is color-coded.

Keywords in **Blue**

Normal text in **Purple**

Comment text in **Black**

Identifier text in **Hot Pink**

The script snippet is indented for ease of use and understanding. Indentation is a very important part of writing code or scripts. It's meaningless to the computer for most languages, but is essential to a person, because it allows you to visually see the structure of a script or code snippet. Script logic (using logical statements like these, if/then/else / etc.) can be quite convoluted. To make it easy to read a script it is advisable that the code be kept as neat as possible.

Note: “If Not...” is a variation on the If conditional statement.

Tips:

Syntax highlighting

Text editor software designed for editing code can use color to differentiate syntax components for ease of readability. This is a feature called syntax highlighting.

Common VBScript conventions

There are a few common script-writing conventions that you can follow to make your scripts easier to read and understand. This is useful if you want to share your scripts, or use the Echoview support service.

- **'Set'**, which is a keyword, is written with its first letter capitalized. VBScript is not case-sensitive, so 'set' or 'sEt' would still work, but capitalization is a common convention.
- Precede a logical grouping of instructions with a comment line that explains what the instructions will do.
- Comment liberally.
- Use indentation to enhance readability.
- **Option Explicit** is a special instruction to the VBScript interpreter, which goes before all other lines of code. It means that you must explicitly declare all of your variables (i.e., use 'Dim'). If you do not use Option Explicit, the script will implicitly assume the declaration and allow you to use a variable that you have not declared. In the example above, we have declared the Echoview application to be EvApp. Without the use of Option Explicit, if you later mistype the variable as EvAppp, a new variable will be created.

Variable lifetime and where a variable is applied

A script can comprise several parts or procedures. A local variable is one that is declared within a part of a script, and it will cease to exist once that procedure is ended. Therefore, you can have several local variables with the same name if they are in different procedures.

If a variable is declared at the start of a script – before the procedures – it will be a global variable and only cease to exist when the script ends.

Windows objects and classes

There are a number of Windows classes and objects available for you to use (for example FileSystemObject). They let you do things like interact with the file system or other computer functions. Advanced scripts can use both Windows and Echoview classes to achieve their goal.

Note: VBS can be particular about when you use brackets for functions and when you don't. For more information, refer to the Microsoft VBScript Language Reference¹¹.

Topic 4 exercises

To reinforce your understanding, the following exercises pose key questions and are accompanied by reasoned answers.

1. Identify the following variable types (object or simple):

Variable	Object type (✓)	Simple type (✓)
Dim a a = "apples"		✓ Simple text variable with quotation marks to indicate text.
Dim h h = 15		✓ Simple number variable.
Dim x Set x = anchovy	✓ Object variables are initialized using the Set keyword.	

2. Is it possible to write a comment and script code on the same line?

Yes. A line can begin with script code and have a comment following, for example:

```
EvApp.OpenFile ("C:\MyFile.ev") 'This opens the ev file
```

You cannot start a line with a comment and follow it with code.

3. What is the difference between a variable and an object?

A variable is a way of referring to an object.

4. Why does a script pause until you click OK when it shows a Message box? How could you use this?

The VBScript command MsgBox is part of the Windows COM standard commands, and will pause your script until you click OK. You can use this to confirm an action before a script moves on to the next section.

5. Can a script variable refer to an Echoview variable?

Yes. A script variable can be used as a way of referring to an Echoview variable.

6. Can you refer to one object in two different ways?

Yes, an object can be referred to in more than one way. Take for example a person called John Smith. You can refer to this person as John and Mr Smith. You are referring to the same person but using different names.

In a script you could have two variables that are, or refer to, the same object. This is related to computer language support for mutable and immutable objects.

Further reading

Microsoft's "Creating Your First Visual Basic Program"¹².

Topic 5: Create a Script for Echoview

In this topic, you can learn how to create your own VBS scripts. Worked examples discuss:

- the processes involved in creating a VBS script for a sequence of Echoview actions
- how to interpret a basic script
- how to correct a syntax error in a script.

Interpreting basic scripts

Several scripts have been provided here, taking you from a very basic interpretation to increasing levels of complexity.

Further examples of scripts may be found in the:

- Echoview Help file: Example scripts for COM automation.
- Echoview website: <https://echoview.com/support/example-scripts/>

Example Script 1: Export integrated regions

Planning a script, and flowchart of steps in Echoview

The data processing phase is complete, and now you are ready to analyze the data. A typical Echoview task is to export integration data for the regions in an EV file. The script for this example is in:

```
C:\Echoview Software\Tutorials\IntroToComScripting\A_single_export.vbs
```

The export procedure for Integration by Regions analysis is supported by the Echoview COM. The time saving for one EV file is trivial but, when applied to many EV files (in a transect or survey), the time savings become significant.

Requirements for the script:

- You have an EV file or folder of EV files that are ready to analyze. File paths and folders are known.
- You want to export Integration by Regions analysis from Echoview.
- You identify the Echoview COM method for the export:

ExportIntegrationByRegionsAll	<p>ExportIntegrationByRegionsAll(string FileName)</p> <p>Summary Export the 'Integration by regions' analysis results for all region classes to the given file name.</p> <p>Parameters</p> <ul style="list-style-type: none"> • FileName The path and name of the file to save results to. <p>Note: This method is an alternative for languages that do not support the optional parameter in ExportIntegrationByRegions.</p> <p>Return True, if successful.</p>
-------------------------------	--

Figure 17. An EvVariableAcoustic method: ExportIntegrationByRegionsAll from the Echoview help file.

Create a flowchart to plan the actions in Echoview and identify the COM objects that are required for the script. The annotated script (next page) also incorporates screen messages for failed conditions.

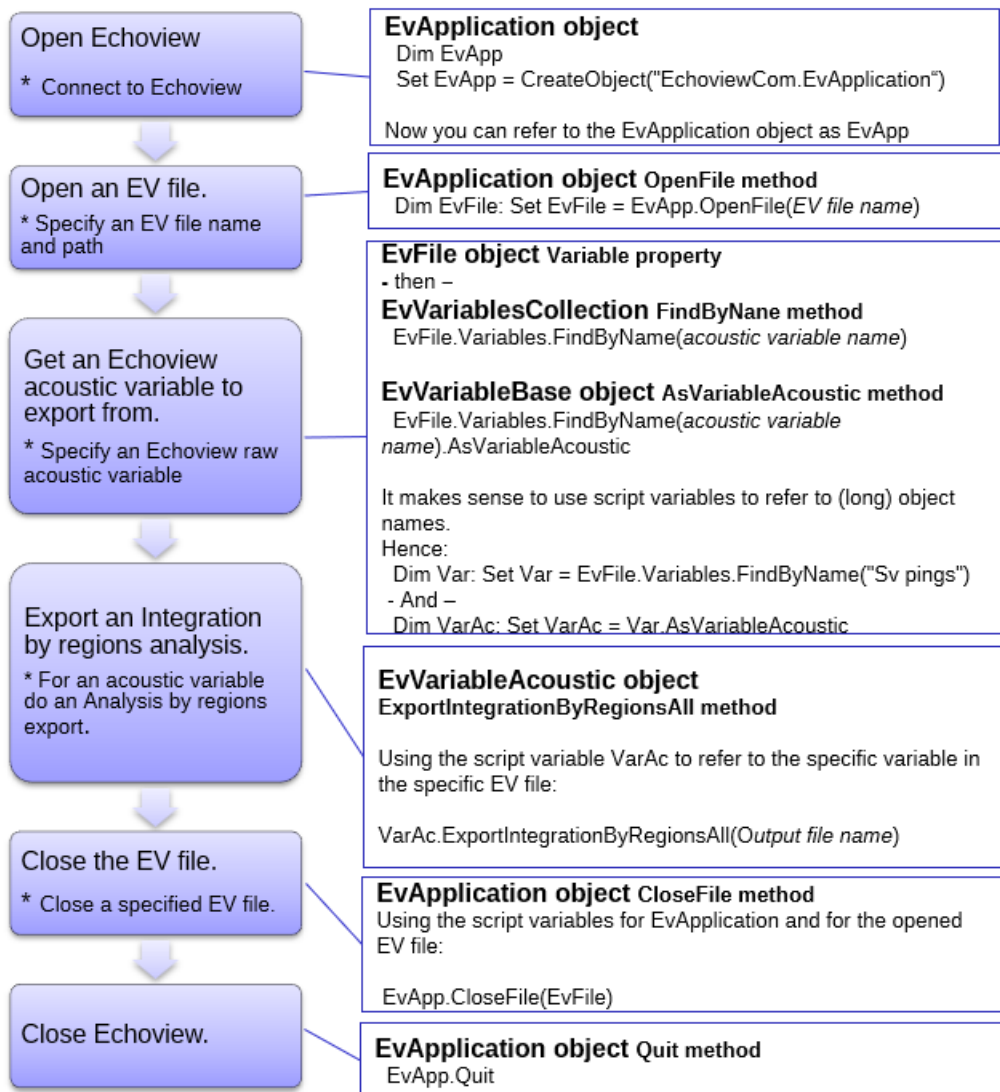


Figure 18. Flowchart relating Echoview steps to sections of a script, *A_single_export.vbs*.

Notes:

You may find that you create, reuse and share a number of scripts. Use techniques such as script comments and script sections to build clear and easy-to-read scripts (for yourself and others).

The Echoview help file page [Example scripts for COM automation](#) describes a number of useful scripts such as:

- Multiple exports from multiple variables.
- Multiple exports from multiple variables in multiple EV files.
- Multiple exports from multiple variables in all EV files in a folder.

A wide variety of scripts are available on the Echoview website: <https://echoview.com/support/example-scripts/>

Line-by-line dissection of a script	VBScript
<p>Give your script a descriptive name</p> <p>Line 2. Prefix by ' to indicate that the following text is a comment.</p>	<pre> 1 2 ' A single export 3</pre>
<p>Open the relevant EV file:</p> <p>Line 4. A comment line for the section.</p> <p>Line 5. Declare a script variable called EvApp and set this to represent your Echoview application.</p> <p>Line 6. Declare a script variable called EvFile and set it to represent COMScripting_Topic4.EV.</p> <p>Line 7. Return a failure message if the file cannot be found.</p> <p>Line 10. Section separator.</p>	<pre> 4 ' Open the EV file 5 Dim EvApp: Set EvApp = CreateObject("EchoviewCom.EvApplication") 6 Dim EvFile: Set EvFile = EvApp.OpenFile("C:\Echoview Software\Tutorials\IntroToComScripting\COMScripting_Topic4.EV") 7 If EvFile Is Nothing Then 8 MsgBox "Failed to open EV file" 9 Else 10</pre>
<p>Open the variable Sv pings</p> <p>Line 11. A comment line for the section.</p> <p>Line 12. Declare a script variable and set it to Sv pings.</p> <p>Line 13. Return a failure message if the variable cannot be found.</p> <p>Line 16. Confirm that it is an acoustic variable.</p> <p>Line 17. Return a failure message if the variable is not acoustic.</p> <p>Line 20. Section separator.</p>	<pre> 11 ' Set the variable 12 Dim Var: Set Var = EvFile.Variables.FindByName("Sv pings") 13 If Var Is Nothing Then 14 MsgBox "Failed to find variable" 15 Else 16 Dim VarAc: Set VarAc = Var.AsVariableAcoustic 17 If VarAc Is Nothing Then 18 MsgBox "Variable is not acoustic" 19 Else 20</pre>

Export the data

Line 21. A comment line for the section.

Line 22. Export all regions integration to FileA-Sv1-ARIN.csv.

Line 23. Return a failure message if the variable is not acoustic.

Line 27. End the script.

Line 28. Section separator.

```
21 | 'Export the variable
22 | If Not VarAc.ExportIntegrationByRegionsAll("C:\Echoview
    | Software\Tutorials\IntroToComScripting\SingleExportOutput\FileA-Sv1-ARIN.csv") Then
23 |     MsgBox "Failed to export by regions"
24 | End If
25 | End If
26 | End If
27 | End If
28 | .....
```

Close the file and Echoview

Line 29. A comment line for the section.

Line 30. Close the EV file.

Line 31. Close Echoview.

```
29 | 'Tidy up
30 | EvApp.CloseFile(EvFile)
31 | EvApp.Quit
32 |
```

Example Script 2: Rename a variable

In this example, you will learn how to interpret a basic script which will:

- start Echoview
- open an EV file
- find the second variable in it. The EV file has five variables. The collection, indexed from 0 – 4, has different data types (acoustic, movement and line variables).
- rename that variable to 'GPS Transect 1'.

The script will pause at each step so you can watch Echoview process each action.

1. Close all instances of Echoview. Open a single instance of Echoview to work with the following example script.

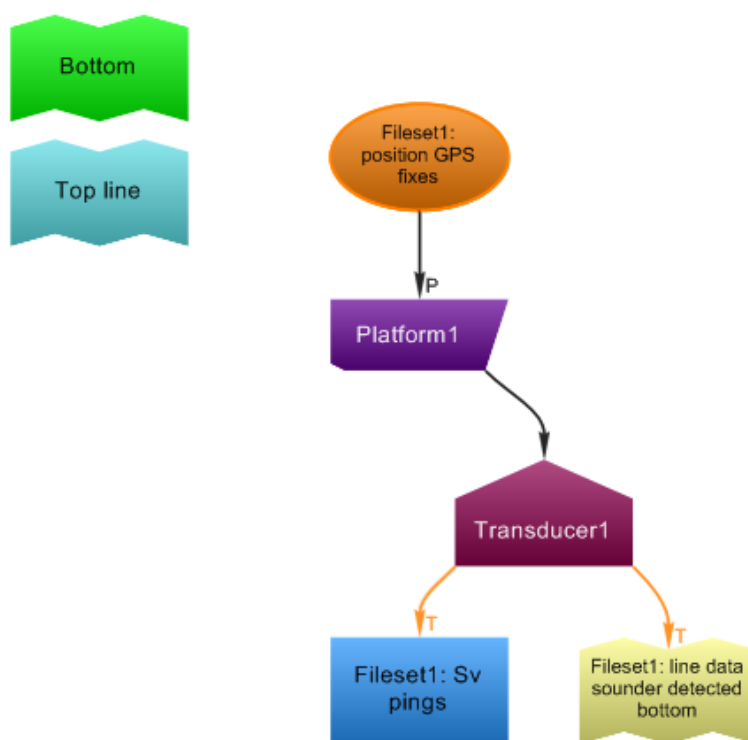


Figure 19. Dataflow from COMScripting_Topic4.EV and the initial name of the GPS variable.

2. Open Windows Explorer.
3. Double-click on the VBScript file called `Rename_variable.vbs` from:

```
C:\Echoview Software\Tutorials\IntroToComScripting\Rename_variable.vbs
```
4. A message box will appear to confirm that Echoview and the requested EV file have been opened. Display the Echoview workspace – this may be hidden by other windows.
5. Click OK.
6. A second message box will appear which identifies a variable in the EV file. Look at the Dataflow window to verify the variable name in the Message box.
7. Click OK.
8. A third message box will appear which indicates that we have changed the name of this variable. Observe the name change in the Dataflow window. Note: this change is not saved to the EV file when the last message box at step 9 is closed. Echoview is also closed at step 9.
9. Click OK.

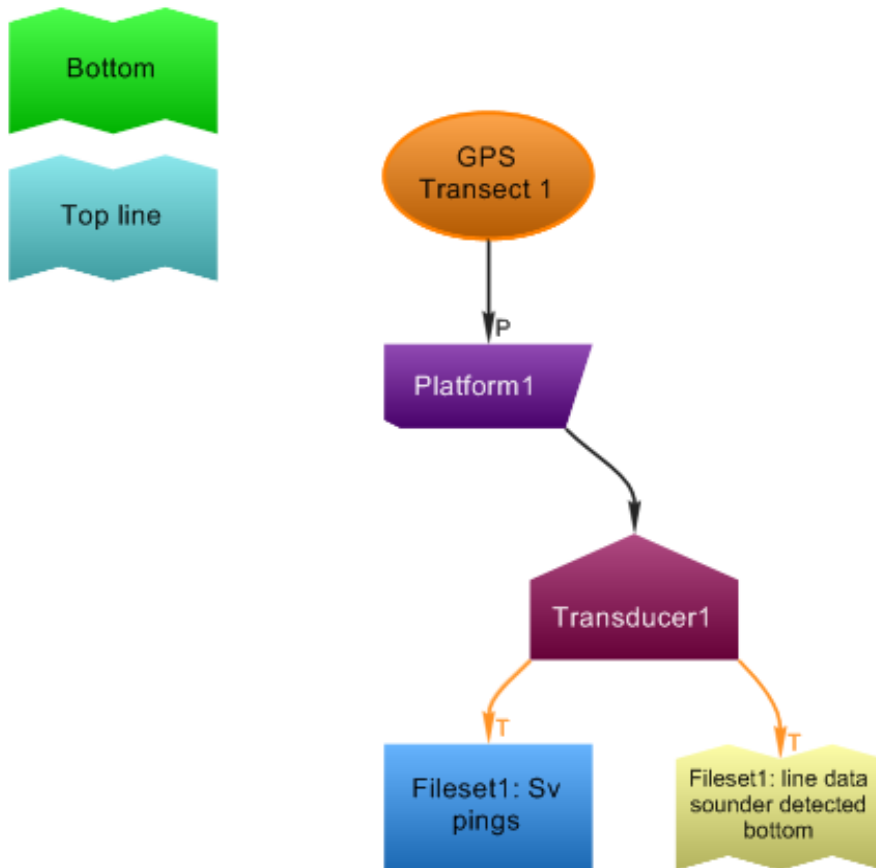


Figure 20. Dataflow with renamed GPS variable.

Examine the `Rename_variable.vbs` script:

1. Start Windows Notepad.
2. Click File, Open.
3. Navigate to the file:
 `C:\Echoview Software\Tutorials\IntroToComScripting\Rename_variable.vbs`
4. Click OK to open the script.

The script for `Rename_variable.vbs` below has important lines in **blue** text.

Comment lines, which explain each instruction, are preceded by a single quotation mark (') and written in **black** text.

Option Explicit

```

' Declare a variable EvApp that is the Echoview application,
' and set it. CreateObject is a Windows COM function that
' finds or starts a "COM server" for us by name.
' "EchoviewCom.EvApplication" is Echoview's name.
Dim EvApp
Set EvApp = CreateObject("EchoviewCom.EvApplication")

' Make a variable for the file we're going to open, and open it
Dim File
Set File = EvApp.OpenFile("C:\Echoview
Software\Tutorials\IntroToComScripting\COMScripting_Topic4.EV")

' Pause
MsgBox "Echoview and the requested EV file are now open"

' Find the second variable, using the collection
' Note that it counts from 0, not 1, so the first variable
' is number 0, and the second number 1.
Dim Var
Set Var = File.Variables(1)

' Pause
MsgBox "Variable 1 in this EV file is " & Var.Name

' Set its name via the Name property
Var.Name = "GPS Transect 1"

' Pause

MsgBox "We have changed the name of Variable 1 - observe the new
name in the Filesets window"

```

Example Script 3: Create EV files for all files in a folder

The following script example creates and saves EV files from all data files in a specified folder.

1. Use Windows Explorer to navigate to:

```
C:\Echoview Software\Tutorials\IntroToComScripting\CreateEVfiles
```

2. Double-click the file `CreateEVfiles.vbs` to run the script.
3. The following message will appear: "Example script to create EV files for all files in a folder".
4. Click OK.
5. A browser window will open and request that you browse to the folder containing data files.
6. Browse to the folder:

```
C:\Echoview Software\Tutorials\IntroToComScripting\CreateEVfiles\Data files
```

7. Click OK.
8. In the "Please enter a number" dialog box, enter the number of data files you would like to add to each EV file.
9. Click OK.

10. In the “Data file extension” dialog box, enter the file extension of the data files you want to add to the EV files. The box will default to ELD files.

11. Click OK.

12. In the “Browse for folder” dialog box, browse to:

```
C:\Echoview Software\Tutorials\IntroToComScripting\
```

13. In the EV File Name dialog box, enter “Script created file – Example 4”

14. Click OK.

15. A message asking whether you would like to use a template file will appear.

16. Click Yes.

17. A dialog box is displayed, and you are requested to enter the full path and name of the template file you would like to use. Enter the following file path:

```
C:\Echoview  
Software\Tutorials\IntroToComScripting\CreateEVfiles\CreateEVfiles_Template.EV
```

18. Click OK.

19. A message box will display the following as confirmation:

Reading files from:

Writing EV files to:

Adding 3 data files per EV file

Using a template:

20. Click OK to end the interaction with the script.

21. Open `CreateEVfiles.vbs` in Windows Notepad to view the script.

Topic 5 exercises

To reinforce your understanding, the following exercises pose key questions and are accompanied by solutions.

Exercise 5.1

1. Open Windows Explorer.
2. Double-click on the VBScript file called `Rename_variable_error.vbs` from:
`C:\Echoview Software\Tutorials\IntroToComScripting\Rename_variable_error.vbs`
3. The script will return an error message (at line 21).
4. Open the file in a script (code) editing program or Windows Notepad.
5. Detect where the error is, correct it and re-save the file. Hint: The error free version would look something like `Rename_variable.vbs`.
6. Run again to ensure that the script now works.

Exercise 5.1 Solution

When you run the script, it will return the following error message:

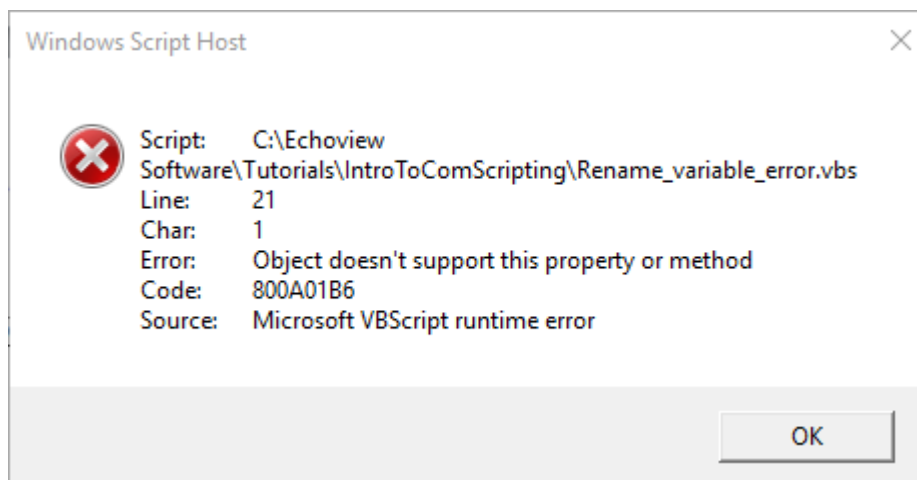


Figure 21. Script error message.

Observe lines 20 and 21 of the script:

The script-variable Var has been dimensioned, but not set.

Insert **Set** at line 21.

Save the file and run again.

```

1 Option Explicit
2
3 ' Declare a variable EvApp that is the Echoview application,
4 ' and set it. CreateObject is a Windows COM function that
5 ' finds or starts a "COM server" for us by name.
6 ' "EchoviewCom.EvApplication" is Echoview's name.
7 Dim EvApp
8 Set EvApp = CreateObject("EchoviewCom.EvApplication")
9
10 ' Make a variable for the file we're going to open, and open it
11 Dim File
12 Set File = EvApp.OpenFile("C:\Echoview
13 Software\Tutorials\IntroToComScripting\COMScripting_Topic4.EV")
14
15 ' Pause
16 MsgBox "Echoview and the requested EV file are now open"
17
18 ' Find the second variable, using the collection
19 ' Note that it counts from 0, not 1, so the first variable
20 ' is number 0, and the second number 1.
21 Dim Var
22 Var = File.Variables(1)
23
24 ' Pause
25 MsgBox "Variable 1 in this EV file is " & Var.Name
26
27 ' Set its name via the Name property
28 Var.Name = "GPS Transect 1"
29
30 ' Pause
31 MsgBox "We have changed the name of Variable 1 - observe the new name in the Filesets window"

```

Figure 22. *Rename_variable_error.vbs*; script displayed using the Notepad++ editor.

Keywords in **Blue**. Numbers in **Red**.

String text in **Grey**

Comment text in **Green**

Identifier text in **Black**.

Exercise 5.2

In this exercise you will fill in the missing parts of the script so that it opens a file, changes schools-detection settings and then detects the schools. The Sv pings echogram has 1270 pings, the potential school regions finish at approximately ping 750. The console command to change the Minimum school length detection property is used in Exec().

1. Start Echoview.
2. Double click on the VBScript file called Schools_detection.vbs from the folder:
C:\Echoview Software\Tutorials\IntroToComScripting\SchoolsDetection
Result: The script will report an error.
3. Open the script in Windows Notepad or a script editing program and enter the correct information where you see the occurrences of ***.
4. Save your changes and run the script again. Display the Sv echogram to view detected schools.

Exercise 5.2 Solution

To successfully run the schools detection, your script should read as follows. Text missing from original file in **bold**:

- look up SchoolsMinimumTotalLength in the Help file to find the property .
- look up DetectSchools in the search function of the Help file to find parameters for schools detection

Option Explicit

' Start Echoview

```
Dim EvApp: Set EvApp = CreateObject("EchoviewCom.EvApplication")
```

' Open the Schools Detection EV file

```
Dim EvFile: Set EvFile = EvApp.OpenFile ("C:\Echoview Software\Tutorials\IntroToCOMScripting\SchoolsDetection\COM_SchoolsDetection.EV")
```

If EvFile Is Nothing Then

```
MsgBox "Failed to open EV file"
```

Else

```
' Change the schools detection settings from the default 40m minimum
' school length to 10m. Use the Echoview Help file to find the
' correct syntax. Note: If using this script for your own data, in
' addition to the schools detections settings, you may need to use
' scripting commands to apply Thresholds on the Data Page, and
' exclusions on the Analysis Page of the Variable Properties dialog
' box. Warning: Using a message box to confirm the change to the
' school length will pause your script until you click OK.
```

```
EvApp.Exec("SchoolsMinimumTotalLength =| 10")
```

```
MsgBox "Minimum school length changed to 10m"
```

'Dimension the variable called Sv Pings

```
Dim Var: Set Var = EvFile.Variables.FindByName("Sv pings")
```

```

If Var Is Nothing Then
    MsgBox "Failed to find variable"
Else
    Dim VarAc: Set VarAc = Var.AsVariableAcoustic
        If VarAc Is Nothing Then
            MsgBox "Variable is not acoustic"
        Else
            ' Perform the schools detection. Note: DetectSchools is a
            ' method of the COM object EvVariableAcoustic. It returns
            ' the number of schools, or -1 if it fails.

            MsgBox VarAc.DetectSchools ("Detectedschools", 1, 750) & " detected schools."&
vbNewline & "-1 means the School detection failed."& vbNewline & "Observe the school
regions. Then press Enter"

                End If
        End If
End If

MsgBox "Press Enter to end this example script, and close Echoview without saving."

```

Topic 6: Viewing Echoview COM Objects and Creating Scripts

Text editor software designed for editing code provide features to:

- browse the COM types embedded in Echoview and of other programs on your computer
- write scripts
- run and debug scripts.

Text editor programs designed for editing code

There are many code editor programs available. The features and capabilities for such programs vary and their cost can vary.

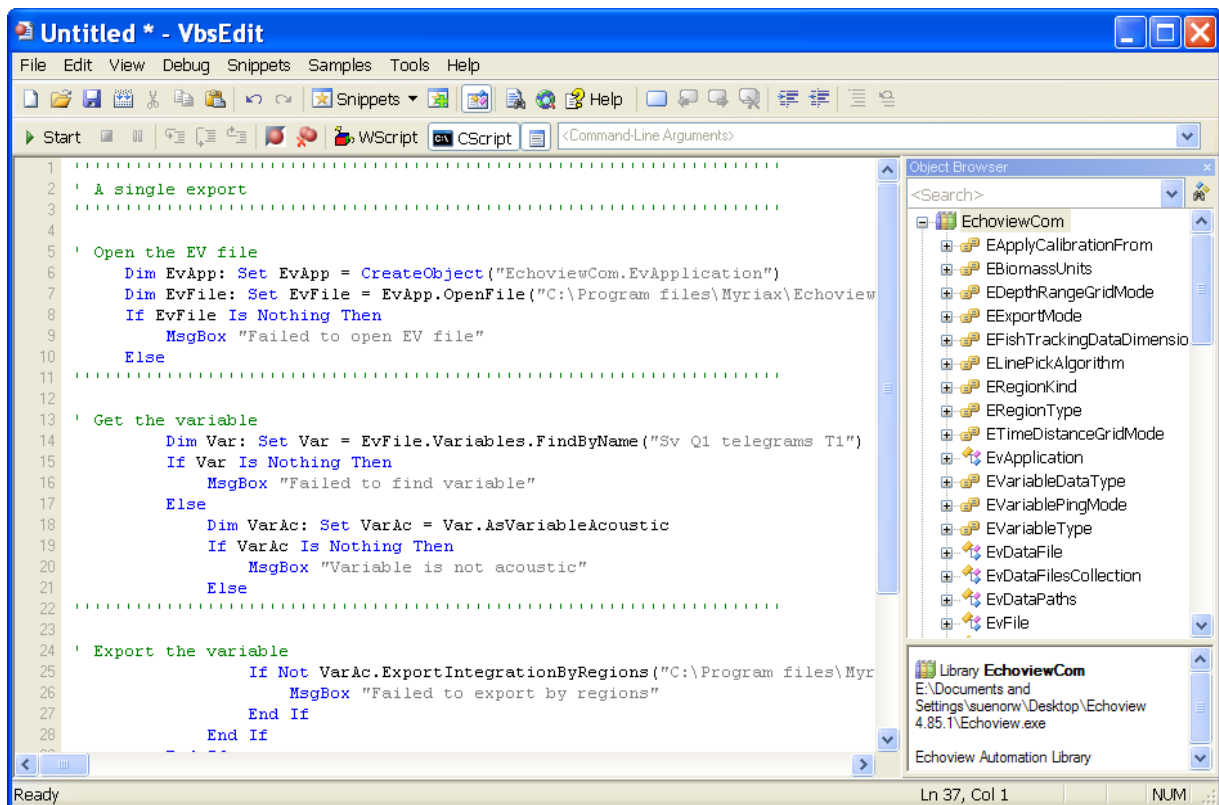


Figure 23. VbsEdit displaying A single export.vbs and the EchoviewCOM Classes. VbsEdit is a program for purchase.

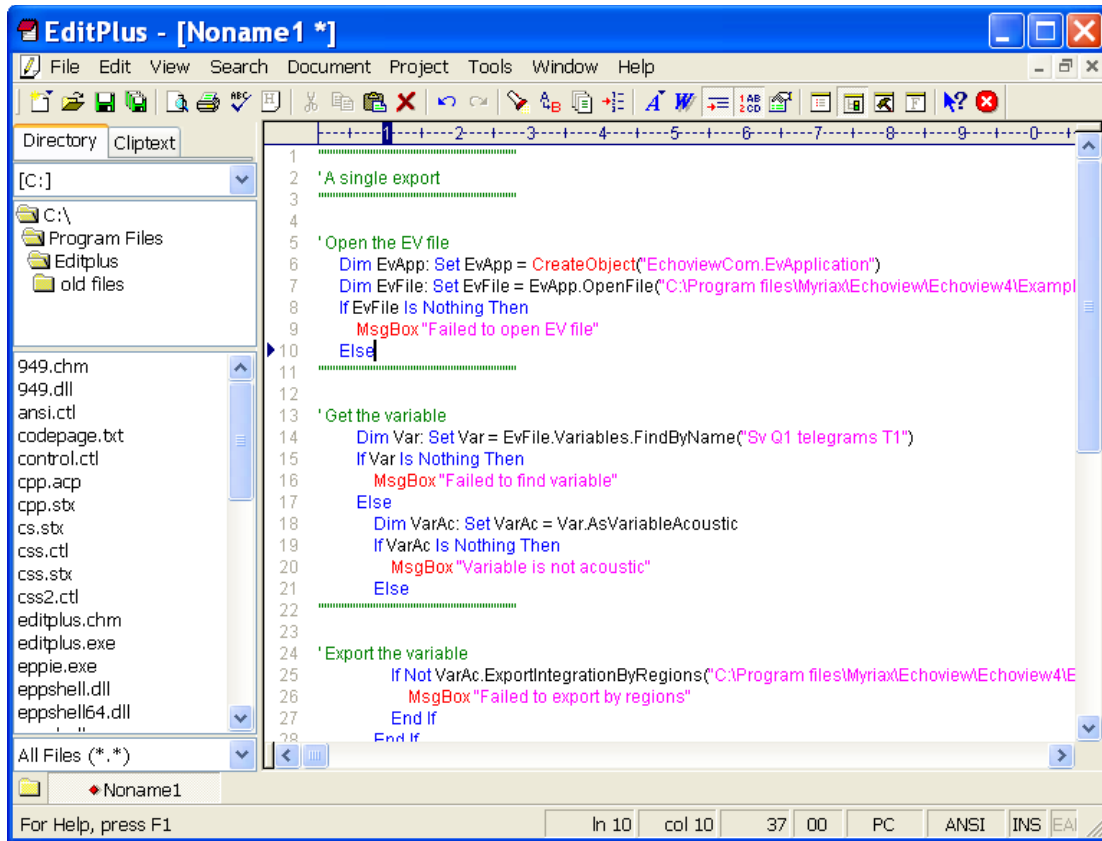


Figure 24. EditPlus displaying A single export.vbs. EditPlus is a program for purchase.

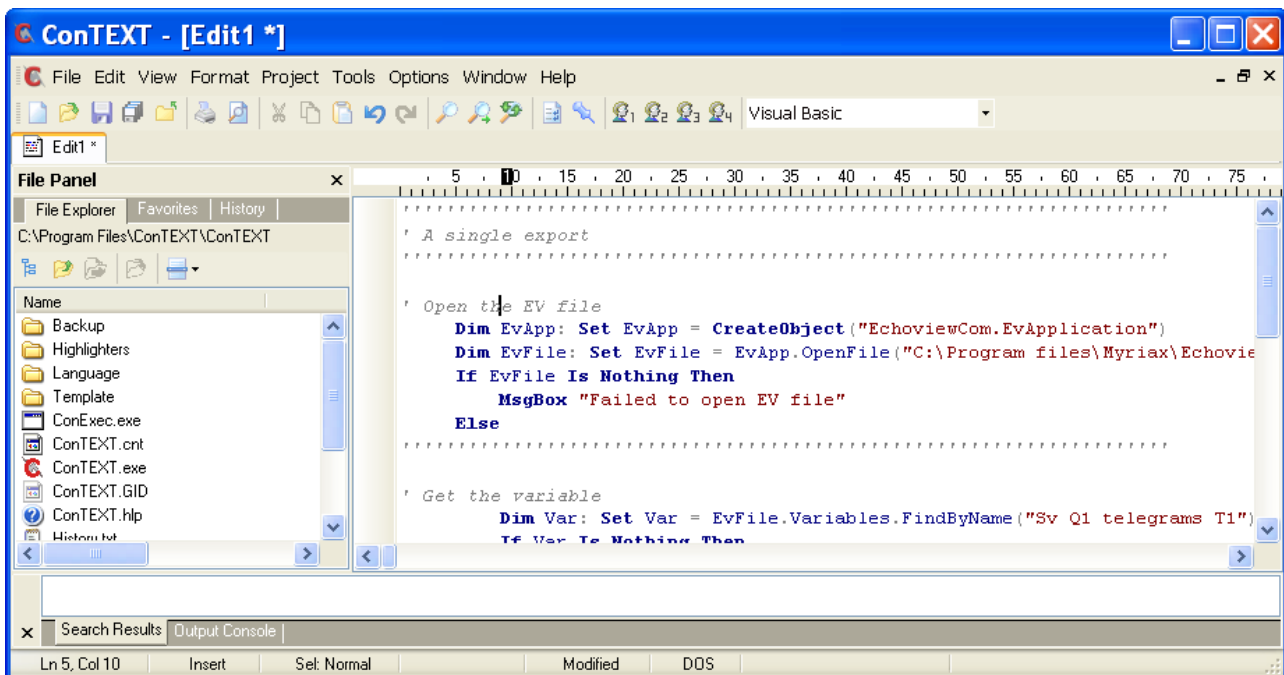


Figure 25. ConTEXT displaying A single export.vbs. ConTEXT is a free program.

Windows Visual Basic editor

Windows programs like Word, Excel or PowerPoint come bundled with a Visual Basic editor that can browse COM classes and offer basic features to write and run scripts. For your version of Office application, search for instructions of how to add the Developer tab to an Word or Excel or PowerPoint.

Opening Microsoft Visual Basic using the Developer tab

1. Open the program (e.g., Word).
2. Click the **Developer** tab.
3. Click **Visual Basic**.
4. On the **Tools** menu, click **References**.
5. On the Reference dialog box, select **Echoview Automation Library**.

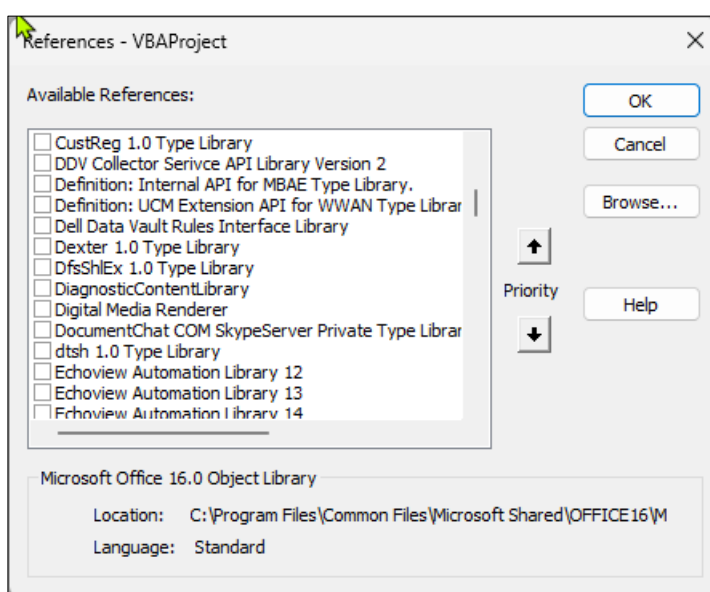


Figure 26. Selecting the Echoview Automation Library

Note: To ensure you have selected the correct Echoview version, refer to the text below the Available references list.

6. Click **OK**.
7. On the **View** menu, click **Object Browser** (F2)
8. On the Object Browser dialog box, select the **EchoviewCom** from the Project/Library list.

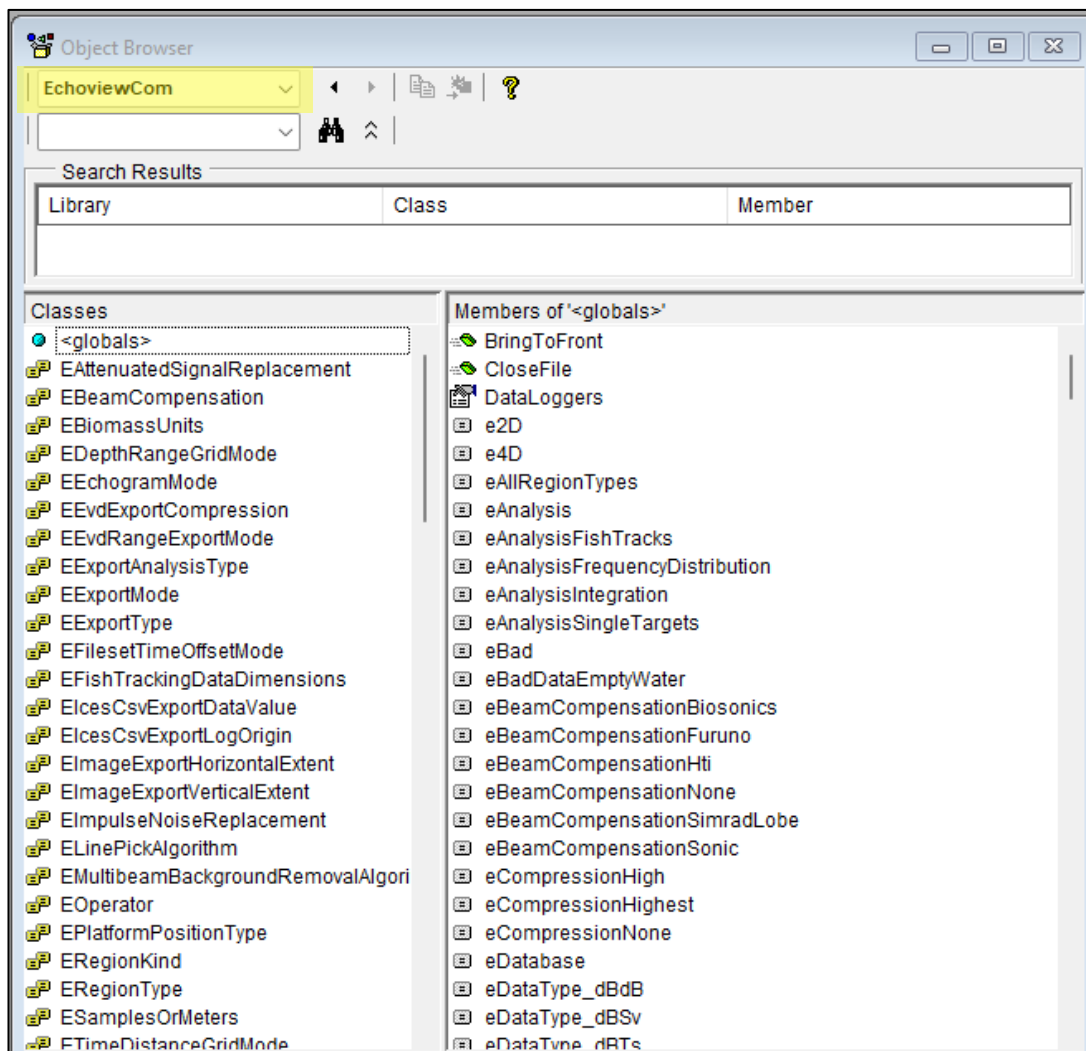


Figure 27. Selecting the Echoview COM object browser

9. The Object Browser will now list the Echoview COM Classes in the left-hand window, and their relevant properties and methods in the right-hand window.
10. Select **EvFile** in the **Classes** section to display its properties and functions in the right-hand side window.

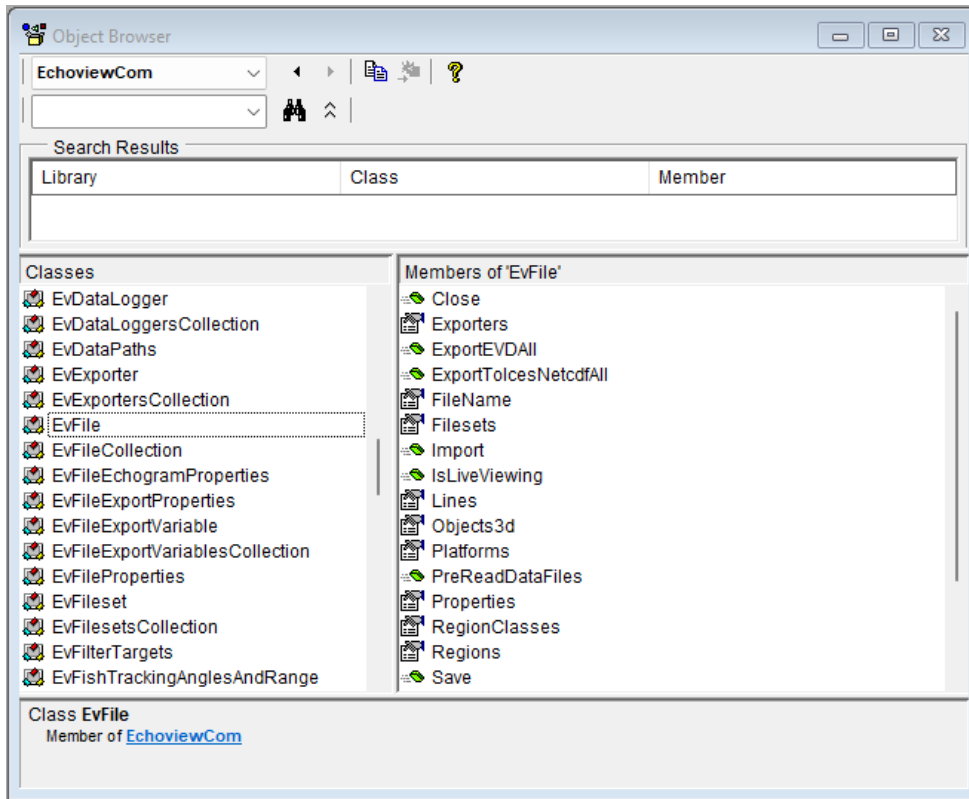


Figure 28. Select the EvFile Class

11. Select the **RegionClasses** Property.

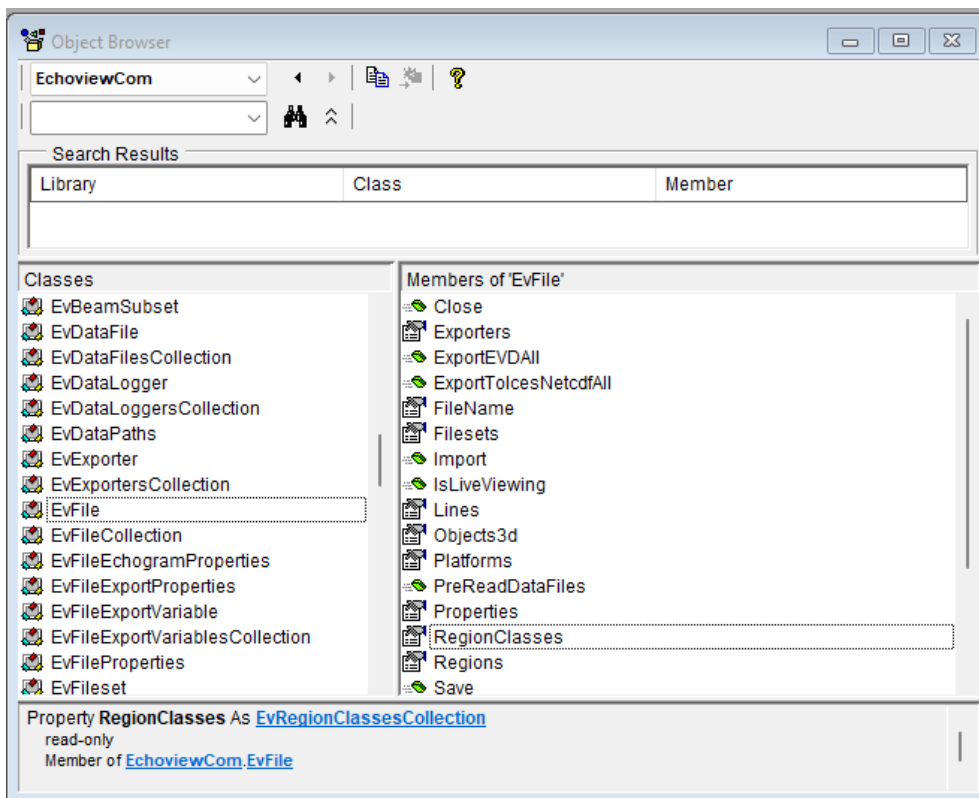


Figure 29. Viewing the RegionClasses property of EvFile.

12. The information window at the bottom displays the Property details.

To write or view scripts – Code window

1. On the **View** menu select **Code**.
2. In the **Code** window paste in the script for **A single export**.
In your tutorial folder, open `A_single_export.vbs` with Notepad, then copy/paste into the Code area.
3. You can set the style and colors for different kinds of script text by using:
Tools menu > **Options** > **Editor Format** tab

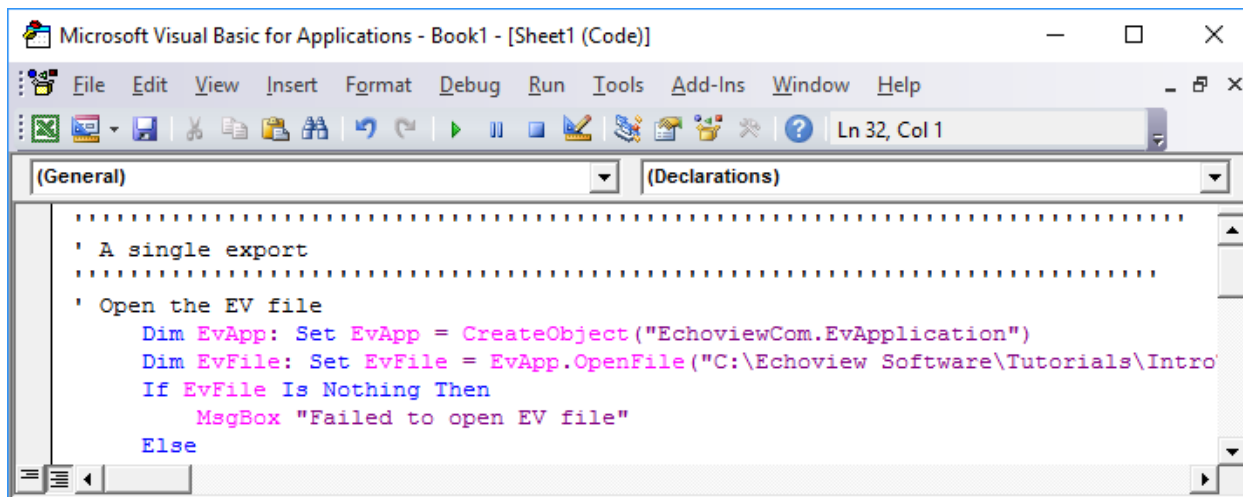


Figure 30. Code window with a script.

4. Close the Microsoft Visual Basic Editor.

References

1. Wikipedia (2024) Visual Basic for Applications [ONLINE] Available at: https://en.wikipedia.org/wiki/Visual_Basic_for_Applications
2. Wikipedia (2024). VBScript. [ONLINE] Available at <https://en.wikipedia.org/wiki/VBScript>
3. Microsoft Documentation (2021) FileSystemObject Reference (Windows Scripting) [ONLINE] Available at: [https://msdn.microsoft.com/en-us/library/hww8txat\(v=vs.84\).aspx](https://msdn.microsoft.com/en-us/library/hww8txat(v=vs.84).aspx)
4. Microsoft (2021) The Component Object Model [ONLINE] Available at: <https://docs.microsoft.com/en-us/windows/desktop/com/the-component-object-model>
5. Microsoft TechNet Script Center (2021) Learn PowerShell Scripting [ONLINE] Available at: <https://technet.microsoft.com/en-us/scriptcenter/dd772284.aspx>
6. Microsoft Developer Network (2021) VBScript Language Reference [ONLINE] Available at: [https://msdn.microsoft.com/en-us/library/d1wf56tt\(v=vs.84\).aspx](https://msdn.microsoft.com/en-us/library/d1wf56tt(v=vs.84).aspx)
7. See also: Microsoft Developer Fabulous Adventures In Coding (2021) VBScript. [ONLINE] Available at: <https://blogs.msdn.microsoft.com/ericlippert/2003/09/15/what-do-you-mean-cannot-use-parentheses/>
8. Microsoft Developer Network (2021) VBScript User's Guide. [ONLINE] Available at: [https://msdn.microsoft.com/en-us/library/sx7b3k7y\(v=VS.85\).aspx](https://msdn.microsoft.com/en-us/library/sx7b3k7y(v=VS.85).aspx)
9. VBScript Tutorials – Herong's Tutorial Examples (2024) Table of Contents. [ONLINE] Available at: <http://www.herongyang.com/VBScript/>
10. Microsoft Developer Network (2021) Programming the FileSystemObject. [ONLINE] Available at: [https://msdn.microsoft.com/en-us/library/2z9ffy99\(v=vs.84\).aspx](https://msdn.microsoft.com/en-us/library/2z9ffy99(v=vs.84).aspx)
11. Microsoft Developer Network (2021) VBScript Language Reference [ONLINE] Available at: [https://msdn.microsoft.com/en-us/library/d1wf56tt\(v=vs.84\).aspx](https://msdn.microsoft.com/en-us/library/d1wf56tt(v=vs.84).aspx)
12. Microsoft Developer Network (2021) Creating Your first Visual Basic Program. [ONLINE] Available at: [https://msdn.microsoft.com/en-us/library/a08t4ke7\(v=vs.100\).aspx](https://msdn.microsoft.com/en-us/library/a08t4ke7(v=vs.100).aspx)

ⁱ Generally, a function is a group of related instructions that take input data and return a value.