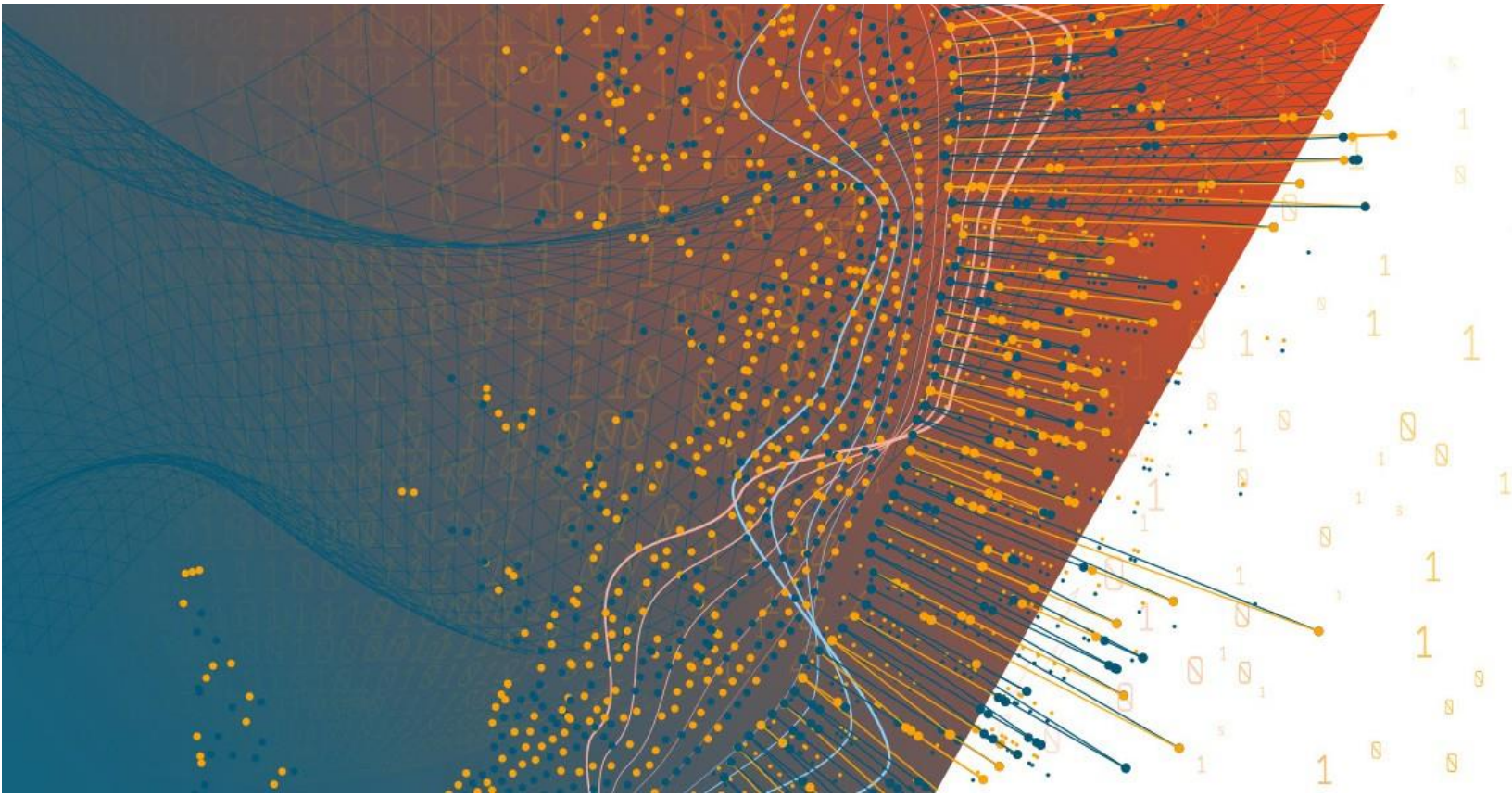

ALTAIR

Altair® Monarch® v2024

PROGRAMMER'S GUIDE

TABLE OF CONTENTS

[1] INTRODUCTION

A number of Monarch customers have asked for a way to integrate Monarch functionality into their own Windows applications. In response to their requests, we created a set of Monarch properties and methods that can be called via COM from languages like C/C++ or Visual Basic. These properties and methods provide programmers with all of the commands necessary to incorporate Monarch's data extraction and export functionality into an application.

Starting with Monarch v12, the automation functionality is contained within a different executable from the interactive application. This removes the very considerable overhead of the Microsoft WPF interface having to load and ensures much faster startup and operation, as well as a lower memory requirement.

For compatibility with existing batch scripts, this executable is called "Monarch.exe", whereas the interactive application is called "DWMonarch.exe".

WHAT IS COM?

COM is an industry standard that applications can use to expose objects to other programs. With COM, applications can allow their objects to be manipulated remotely, via program control. The application that is providing the object creates and manages it. The controlling application manipulates the object by setting properties and performing methods (i.e. actions) through program method calls.

Advantages of Using COM

Even without COM, any Windows application can be run [launched] from another application by employing the WinExec function. However, a program launched in this manner will run independently of the application which launched it. You cannot be certain that the launched application will finish executing before the next statement in your main application is processed. Furthermore, there is no link between the programs. If your main program terminates, the program that it launched may continue to run. There is no way to say, "If the main program shuts down, turn off any other programs it launched".

By using COM, these problems can be eliminated. Monarch is established as a COM server and the main program becomes the client, using method calls to communicate with the server. Execution of a statement in the calling program will not occur until the previous method has been completed. When the calling program terminates, the server is shut down, closing down the Monarch application.

Another advantage of using COM becomes evident when you have an application that requires automation of multiple passes through Monarch. Using the WinExec (or the Monarch command line) method, the entire Monarch application, report file, and model must be loaded for each pass. With COM, the Monarch COM server is loaded only once. You can apply a new sort or filter, load a new model, or load a new report through method calls, resulting in significantly faster execution.

[2] MONARCH PROPERTIES AND METHODS

This chapter describes each of the Monarch methods and properties. It is important to note that only a subset of the functionality of Monarch can be accessed via COM. Monarch's methods and properties provide commands necessary to launch and control a Monarch export session from another application. This subset provides the ability to perform the following tasks.

To begin, launch Monarch as a COM Server process.

This capability allows the automation of Monarch's data extraction and export capabilities. The following tasks apply to the use of Monarch in this role:

- Open a report or a series of reports. Monarch can extract data from any number of reports at-a-time. For information about extracting data from a series of reports, see the section entitled Opening Multiple Instances of a Report in the Monarch Help file
- Open a model file.
- Query the model file to determine what filter, sort, and summary definitions are available.
- Select a filter definition from the model file to apply to the extracted data set.
- Select a sort definition from the model file to apply to the extracted data set.
- Select a summary definition from the model file to generate a summary report from the extracted data set.
- Export the extracted data set or the summary to any of Monarch's supported file formats (see the Monarch Help file for a complete description of each supported export format).

PROPERTIES

CURRENTFILTER (String)

CurrentFilter is a variable that is used to set or query the name of the currently active filter definition. CurrentFilter accepts a string of up to 31 characters representing the name of the currently active filter definition.

When a model file is opened (via the SetModelFile method), CurrentFilter is set to the name of the active filter definition established in the model. When a model is opened as part of a project file (via the SetProjectFile method), CurrentFilter is set to the name of the active filter definition referenced in the project file. If no filter definition is referenced in the project file, the active filter definition from the model file is used. If no active filter definition is established in the project or the model, the default value of CurrentFilter is blank (an empty string).

This variable may also be used to establish an active filter definition or change the currently active filter definition. To do this, simply assign it the name of one of the filter definitions stored in the model. If the new name assigned does not match any of the values stored in the model, the value of CurrentFilter remains unchanged.

CURRENTMODEL (String)

CurrentModel is a variable that is used to set or query the name of a currently active model. CurrentModel accepts a string of up to 256 characters representing the name of the currently active model.

CURRENTSORT (String)

CurrentSort is a variable that is used to set or query the name of the currently active sort definition. CurrentSort accepts a string of up to 31 characters representing the name of the currently active sort definition. When a model file is opened (via the SetModelFile method), CurrentSort is set to the name of the active sort definition established in the model. When a model is opened as part of a project file (via the SetProjectFile method), CurrentSort is set to the name of the active sort definition referenced in the project file. If no sort definition is referenced in the project file, the active sort definition from the model file is used. If no active sort definition is established in the project or the model, the default value of CurrentSort is blank (an empty string).

This variable may also be used to establish an active sort definition or change the currently active sort definition. To do this, simply assign it the name of one of the sort definitions stored in the model. If the new name assigned does not match any of the values stored in the model, the value of CurrentSort remains unchanged.

CURRENTSUMMARY (String)

CurrentSummary is a variable that is used to set or query the name of the currently active summary definition. CurrentSummary accepts a string of up to 31 characters representing the name of the currently active summary definition.

When a model file is opened (via the SetModelFile method), CurrentSummary is set to the name of the active summary definition established in the model. When a model is opened as part of a project file (via the SetProjectFile method), CurrentSummary is set to the name of the active summary definition referenced in the project file. If no summary definition is referenced in the project file, the active summary definition from the model file is used. If no active summary definition is established in the project or the model, the default value of CurrentSummary is blank (an empty string).

This variable may also be used to establish an active summary definition or change the currently active summary definition. To do this, simply assign it the name of one of the summary definitions stored in the model. If the new name assigned does not match any of the values stored in the model, the value of CurrentSummary remains unchanged.

EXPORTS (String Array)

Exports is a string collection containing all project exports in the currently open project file. If the project contains no export definitions or if no project file is currently open, the collection is empty.

FILTERS (String Array)

Filters is a string collection containing all filters available in the currently open model file. If the model contains no filter definitions or if no model file is currently open, the collection is empty.

This collection was introduced in Monarch v12.

FILTERCOUNT (Integer)

FilterCount returns the total number of filter definitions available in the currently open model file. If the model contains no filter definitions or if no model file is currently open, the method returns 0.

MESSAGES (STRING ARRAY)

Messages is a string collection containing the most recent error or informational message.

This collection was introduced in Monarch v12.

SORTCOUNT (Integer)

SortCount returns the total number of sort definitions available in the currently open model file. If the model contains no sort definitions or if no model file is currently open, the method returns 0.

SORTS (String Array)

Sorts is a collection containing all sorts available in the currently open model file. If the model contains no sort definitions or if no model file is currently open, the collection is empty.

This collection was introduced in Monarch v12.

SUMMARIES (String Array)

Summaries is a string collection containing all summaries available in the currently open model file. If the model contains no summary definitions or if no model file is currently open, the collection is empty.

This collection was introduced in Monarch v12.

SUMMARYCOUNT (Integer)

SummaryCount returns the total number of summary definitions available in the currently open model file. If the model contains no summary definitions or if no model file is currently open, the method returns 0.

METHODS

CLOSEALLDOCUMENTS

CloseAllDocuments closes all open report files and the model file. It also closes the project file used.

EXIT

Exit closes all open report files and the associated model file. It also closes the project file used, as well as the log file, and terminates the Monarch session (removing the Monarch COM server from memory).

EXPORTSUMMARY(STRING *EXPORTFILE*)

ExportSummary causes the data to be exported from the Summary view and written to the file *export file*. If the full file name (including drive and path) is not supplied, the default Export Files location stored in the Monarch defaults will be used

The export file type is determined by the file extension specified for *export file* (for a list of supported export file formats and their corresponding file extensions, see the Monarch Help file). If the file extension is not provided, the default Export File extension in the Windows Registry will be used.

Return Value: Boolean TRUE if the export completed successfully, otherwise returns FALSE.

If the file cannot be written, an error message will be written out to the log file (in the folder specified via the Monarch logging settings). The log file will be named according to the following convention:

Monarch-YYYY-MM-DD-HH-MM-SS-FS

(Where FS is fractional seconds ranging from 0-99)

EXPORTTABLE(STRING *EXPORTFILE*)

ExportTable causes the data to be exported from the Table view and written to the file *export file*. If the full file name (including drive and path) is not supplied, the default Export Files location stored in the Monarch defaults will be used. If the file extension is not provided, the default Export File extension in the Windows Registry will be used. Only those records matching the **CurrentFilter** will be exported to the file. Records are exported in the order specified by **CurrentSort**.

The export file type is determined by the file extension specified for *export file* (for a list of supported export file formats and their corresponding file extensions, see the Monarch Help file).

Return Value: Boolean TRUE if the export completed successfully, otherwise returns FALSE.

If the file cannot be written, an error message will be written out to the log file (in the folder specified via the Monarch logging settings). The log file will be named according to the following convention:

Monarch-YYYY-MM-DD-HH-MM-SS-FS

(Where FS is fractional seconds ranging from 0-99)

GETFILTERNAMEAT(INTEGER *INDEX*)

GetFilterNameAt returns the name of the *n*th filter from within the currently open model. The parameter *n* must be a value between 0 and FilterCount-1 (FilterCount is the value returned by the **FilterCount** method). If the value of *n* is not within the legal range, or if no model is open or if there are no filter definitions established for the currently open model, the method returns an empty string. Otherwise, the method will return a text string (up to 31 characters) representing the name of the specified filter.

GETSORTNAMEAT(INTEGER *INDEX*)

GetSortNameAt returns the name of the *n*th sort definition in the currently open model. The parameter n must be a value between 0 and SortCount-1 (SortCount is the value returned by the **SortCount** method). If the value of *n* is not within the legal range, or if no model is open or if there are no sort definitions established for the currently open model, the method returns an empty string. Otherwise, the method returns a text string (up to 31 characters) representing the name of the specified sort definition.

GETSUMMARYNAMEAT(INTEGER *INDEX*)

GetSummaryNameAt returns the name of the *n*th summary definition in the currently open model. The parameter *n* should be a value between 0 and SummaryCount-1 (SummaryCount is the value returned by the **SummaryCount** method). If the value of *n* is not within the legal range, or if no model is open or if there are no summary definitions established for the currently open model, the method returns an empty string. Otherwise, the method returns a text string (up to 31 characters) representing the name of the specified summary definition.

ISACTIVE

IsActive queries the COM server to determine whether the Monarch COM server is active. If the Monarch server is active, this method returns a positive integer. If the server is not active, an error condition occurs, which must be trapped. Refer to the programming example in Chapter 2 for sample code illustrating the use of this method.

JETEXPORTSUMMARY(STRING EXPORTFILE, STRING TABLENAME, INTEGER APPENDFLAG)

JetExportSummary causes the data to be exported from the Summary view and written to the file *ExportFile*, to the table specified by *TableName.* Valid *AppendFlag* values are 0 for overwrite, 1 for new table or sheet and 2 for append to existing table or sheet. For some formats, it is not possible to specify a table or sheet name and the *TableName* will be ignored. Additionally for these cases, multi table or sheet options are not allowed.

The export file version is defined by the settings in the Monarch Options under Folders & File Types and will apply to the extension specified as part of *ExportFile*.

Return Value: Boolean TRUE if the export completed successfully, otherwise returns FALSE.

If the file cannot be written, an error message will be written out to the log file (in the folder specified via the Monarch logging settings). The log file will be named according to the following convention:

Monarch-YYYY-MM-DD-HH-MM-SS-FS

(Where FS is fractional seconds ranging from 0-99)

JETEXPORTTABLE(STRING *EXPORTFILE,*STRING *TABLENAME*, INTEGER *APPENDFLAG*)

JetExportTable causes the data to be exported from the Table window and written to the file *ExportFile*, to the table specified by *TableName.* Valid *AppendFlag* values are 0 for overwrite, 1 for new table or sheet and 2 for append to existing table or sheet. For some formats, it is not possible to specify a table or sheet name and the *TableName* will be ignored. Additionally for these cases, multi table or sheet options are not allowed.

The export file version is defined by the settings in the Monarch *Options* dialog under Export Options and will apply to the extension specified as part of *ExportFile*.

Return Value: Boolean TRUE if the export completed successfully, otherwise returns FALSE.

If the file cannot be written, an error message will be written out to the log file (in the folder specified via the Monarch logging settings). The log file will be named according to the following convention:

Monarch-YYYY-MM-DD-HH-MM-SS-FS

(Where FS is fractional seconds ranging from 0-99)

OPENDATABASE(STRING CONNECTSTRING,STRING PASSWORD, STRING TABLE|VIEW,STRING MODEL)

OpenDatabase method opens an ISAM database file or an ODBC data source. This method is used in place of the **SetReportFile** method, which is used to open one or more report files. An automated Monarch session cannot open both the **OpenDatabase** and **SetReportFile** methods, as the Monarch Table window may be populated from only a single source, either a database or a series of reports.

ConnectString is either a string representing the entire ODBC connection string or a string containing the path to the data source on the network or the local drive. *Password* is a string that can be up to 32 characters long which is used to open a password protected data source.

Table|*view* is the name of the table or view from which to import data. This parameter is not used when opening an ISAM file that does not support multiple tables. *Model* is the path and file name of the model file to use for the session.

A model is required for this method. The model contains parameters needed to complete the database import. If the model also contains join parameters specifying a join to a password protected ODBC source, a **SetJoinPassword** and **SetDatabaSource** method could be issued after opening the model file.

RUNALLEXPORTS()

RunAllExports executes all Project Exports in the currently loaded Project.

Note that a call to SetProjectFile must have been made previously and a Project Export of the specified name must exist.

Return Value: Boolean TRUE if the export was performed successfully, otherwise returns FALSE.

RUNEXPORT(STRING *EXPORTNAME*)

RunExport executes a Project Export of the name specified in ExportName in the currently loaded Project.

Note that a call to SetProjectFile must have been made previously and a Project Export of the specified name must exist.

Return Value: Boolean TRUE if the export was performed successfully, otherwise returns FALSE.

SETDATASOURCEPASSWORD(STRING *LOOKUPNAME*,STRING *PASS WORD*)

SetDataSourcePassword sets the password for an external lookup specified by LookupName. More than one call to **SetDataSourcePassword** can be made to specify password for each lookup associated with a model. This method must be called before calling **SetModelFile.**

Unlike the single password set with **SetJoinPassword**, passwords set with **SetDataSourcePassword** apply only to the very next model load, after which they are discarded. A password set with **SetDataSourcePassword** takes precedence over one set with **SetJoinPassword**. A password set with **SetDataSourcePassword** always gets applied, even if it's empty.

SETINPUTCHARACTERSET(STRING *CHARACTERSET*)

SetInputCharacterSet establishes the character set for interpreting report data that is loaded via the **SetReportFile** method.

CharacterSet is a string value of one of the following allowed values:

"ANSI"

"ASCII"

"UTF-8"

"UTF16-LE"

"UTF16-BE"

SETMODELFILE(STRING *MODELFILE*)

SetModelFile opens the specified *model file* for processing. If the full file name (including drive and path) is not supplied, the default Model file location stored in the Monarch defaults will be used. If the file is successfully opened, the method returns a value of true. Otherwise, it returns false. If a model file is already open when the **SetModelFile** method is called, the previously opened model is closed before the new model is opened.

SETJOINPASSWORD(STRING *PASSWORD)*

SetJoinPassword establishes the password to be used when opening a password-protected data source. Password is a string that can be up to 32 characters long. A **SetModelFile** method or an **OpenDatabase** method should follow this method.

SETOUTPUTCHARACTERSET(STRING *CHARACTERSET*)

SetOutputCharacterSet establishes the character set for exporting data to text and delimited text formats.

CharacterSet is a string value of one of the following allowed values:

“ANSI”

“ASCII

“UTF-8”

“UTF16-LE”

“UTF16-BE”

SETPASSWORDPROTECTEDREPORTFILE(STRING *REPORTNAME*, STRING *PASSWORD*, BOOL *APPEND*)

SetPasswordProtectedReportFile opens the specified PDF report file with the specified password for processing. If *append* is true (non-zero), the report file is added to the list of open reports. There is no limit to the number of reports that may be open at one time. If *append* is false (zero), all previously opened reports are closed before the new report file is opened.

If the file is successfully opened, the method returns a value of true. Otherwise, it returns false.

SETPROJECTFILE(STRING *PROJECTFILE*)

SetProjectFile opens the specified project file for processing. If the full file name (including drive and path) is not supplied, the default Published Files location stored in the Monarch defaults will be used. If the report file(s) and model file referenced by the project file are all successfully opened, the method returns a value of true. Otherwise, it returns a value of false. Any previously opened report or model files are closed before the **SetProjectFile** method is executed.

SETREPORTFILE(STRING *REPORTFILE,*BOOLEAN APPEND)

SetReportFile opens the specified *report file* for processing. If the full file name (including drive and path) is not supplied, the default Report Files location stored in the Monarch defaults will be used. If *append* is true (non-zero), the report file is added to the list of open reports. There is no limit to the number of reports that may be opened at one time. If *append* is false (zero), all previously opened reports are closed before the new report file is opened.

If the file is successfully opened, the method returns a value of true. Otherwise, it returns false.

SETRUNTIMEPARAMETER(STRING *FIELDNAME,* STRING *FIELDVALUE*)

SetRuntimeParameter allows the setting of a runtime parameter value of the Runtime Parameter field specified by FieldName to the value specified in FieldValue.

SetRuntimeParameter may be called either before or after a model is loaded.

If SetRuntimeParameter is called before loading the model, then the value is cached so that it becomes effective at the next model load. Multiple calls may be made so that one can set all the runtime parameters for a particular model before actually loading the model, and thereby suppress the Runtime Parameters dialog that would otherwise pop up at model load time.

Calls to SetRuntimeParameter that occur before loading a model are always successful, since there is no field list against which to test field names or types.

If SetRuntimeParameter is called after a model is loaded, the new value thus provided takes effect immediately. Such calls will fail if the specified field name can't be found in the model, or if the specified value cannot be converted to the proper data type.

Return Value: Boolean TRUE if the file was parameter was set successfully, otherwise returns FALSE.

SETTEXTAPPEND(BOOLEAN *APPEND*)

SetTextAppend determines whether Monarch will overwrite or append to an existing text or delimited text file. This method is called prior to calling the **ExportTable** or **ExportSummary** method. If *Append* is 0, the existing text or delimited text file will be overwritten. If *Append* is 1, data will be appended to an existing file.

VERSION

Version returns the Monarch version number as a text string in the form "Version X.X.X.X".

WRITETOLOGFILE(STRING *USERLOGINFO*)

WriteToLogFile writes the specified *string* to the log file.

[3] USING MONARCH COM AUTOMATION

MONARCH COM REGISTRATION

When Monarch is installed, it registers itself as a COM server. Once registered, the server becomes available for COM transactions.

It is also possible to perform the COM registration manually. Using an administrative command prompt, type "Monarch.exe /register" in the path where the Monarch.exe executable resides.

The COM server name for Monarch is **Monarch32**.

CALLING THE MONARCH COM SERVER FROM A CLIENT APPLICATION

Before the client application can use any of the Monarch methods, it must create the COM object that will contain pointers to the Monarch COM server. Once this object has been created, the client application has control of Monarch.

In Visual Basic, the statement used to create the object would take the form

Set MonarchObj=CreateObject("Monarch32")

Once the object has been created, you may use any of the Monarch methods.

Program Subroutine Example

A sample Visual Basic for Applications subroutine that invokes Monarch, opens a report and model, applies a couple of filters, and exports the resulting table to Excel is provided below:

Note

```
Private Sub Monarch_Process()
Dim MonarchObj As Object
Dim openfile, openmod As Boolean

Const ReportFolder = "C:\Users\Public\Documents\Datawatch
Monarch\Reports\"
Const ModelFolder = "C:\Users\Public\Documents\Datawatch
Monarch\Models\"
Const ExportFolder = "C:\Users\Public\Documents\Datawatch
Monarch\Export\"

'If a Monarch COM server is currently active, GetObject will use it.
'If it is not, use the CreateObject() to create one.
```

```
Set MonarchObj = GetObject("", "Monarch32")

If MonarchObj Is Nothing Then
    Set MonarchObj = CreateObject("Monarch32")
End If

openfile = MonarchObj.SetReportFile(ReportFolder & "classic.prn",
False)

openmod = MonarchObj.SetModelFile(ModelFolder & "Lesson9.dmod")

If openfile = True And openmod = True Then
'Set filter and export to Excel XLS
    MonarchObj.CurrentFilter = "Fandangos Records"
    MonarchObj.ExportTable (ExportFolder & "Fandangos.xls")
    MonarchObj.CurrentFilter = "No Returns"
    MonarchObj.ExportTable (ExportFolder & "No_Returns.xls")
Else

'Error handling here

End If

MonarchObj.CloseAllDocuments

MonarchObj.Exit

Set MonarchObj = Nothing

End Sub
```

[APPENDIX A] ERROR MESSAGES

OLE AUTOMATION SERVER CANNOT CREATE OBJECT

This error message is issued when the Monarch COM routines are inaccessible to the calling program. This error will occur if the routines have not been registered or are already in use. You can check that the routines are registered by running REGEDIT.EXE and examining the list of program registrations. If another program is using the routines, that program must either terminate or issue an Exit command to release the COM routines.

You may also want to check that Monarch is installed properly, or attempt to perform the COM registration manually by opening an administrative command prompt in the same folder as the Monarch.exe executable and issuing the following command:

```
Monarch.exe /register
```