[The Complete] Management Solution
For Your Network

PRODUCT MANUAL

ManageWise™ 2.5

Desktop Management OLE API Guide

ManageWise™
MANAGEMENT SOFTWARE

Novell®

disclaimer

Novell, Inc. makes no representations or warranties with respect to the contents or use of this manual, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any NetWare software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of NetWare software, at any time, without any obligation to notify any person or entity of such changes.

trademarks

Novell, Inc. has attempted to supply trademark information about company names, products, and services mentioned in this manual. The following list of trademarks was derived from various sources.

Novell and NetWare are registered trademarks of Novell, Inc. in the United States and other countries.

The Novell Network Symbol, Internet Packet Exchange, IPX, IPX/SPX, ManageWise, NDS, NetWare 3, NetWare 4, NetWare Directory Services, NetWare DOS Requester, NetWare Loadable Module, NetWare MHS, NetWare Message Handling System, NLM, Novell Directory Services, Sequenced Packet Exchange, SMS, SPX, Virtual Loadable Module, and VLM are trademarks of Novell, Inc.

Apple, AppleLink, AppleTalk, LocalTalk, Macintosh, and Power Macintosh are registered trademarks and Apple Desktop Bus is a trademark of Apple Computer, Inc. CompuServe is a registered trademark of CompuServe Incorporated. NetPort and SatisFAXtion are registered trademarks and

Novell, Inc.
2180 Fortune Drive
San Jose, CA 95131

ManageWise™ 2.5
Desktop Management OLE API
October 1997
100-003624-001B

Contents

chapter 1

Using the ManageWise OLE API

The following ManageWise™ components are enabled for Object ref-headLinking and Embedding (OLE):

- Desktop Manager
- Queue Monitor

Each ManageWise component provides a set of features that you can access using a BASIC-style programming language, such as Microsoft Visual Basic. For example, one of the features associated with Desktop Manager is File Transfer. The Desktop Manager OLE API provides a call that initiates a session between the local workstation and a remote workstation.

OLE Automation

OLE Automation is a process that enables you to access an application's features. These features are made available through a hierarchy of *OLE Automation objects*. The way in which you access the features of a ManageWise component depends on the object hierarchy of that component.

Associated with each object are two attributes that determine how the object can be used:

- Methods, which perform a particular action on an object
- Properties, which set or return information about the state of an object

Object Hierarchy

You access the features of each OLE-enabled ManageWise component through a prescribed object hierarchy. In general, this hierarchy is similar to the following example:

```
Application object
Collection object 1
   Object 1
Collection object 2
   Object 2
```

The hierarchical structure of the objects reflects the features of the corresponding component. For a description of the ManageWise Desktop Manager components, see *ManageWise 2.5 Desktop Management Guide*.

Object Attributes

Each OLE API chapter provides a table similar to the following that lists the objects and attributes (methods and properties) that are available for each component.

Objects	Methods	Properties
Application Object	Method 1 Method 2	Property 1 Property 2 Property 3 Property 4
Object 1	Method 3 Method 4	Property 5 Property 6 Property 7

Obtaining Member Objects from Collection Objects

Some ManageWise components have OLE objects categorized as collections. For example, Desktop Manager has Workstations, Servers, Aliases, and Groups collection objects. A *collection object* is a container that holds individual objects called *members*. A member object is typically called by the singular form of the collection object's name. For example, the Workstations collection object has member Workstation objects.

You obtain a member from a collection by specifying the member as a string or integer in parentheses following the collection object's name as shown in the following example:

CollectionObject(*Identifier*)

CollectionObject represents a collection object, such as Workstations. *Identifier* represents the member, which can be a string or an integer.

If *Identifier* is a string, the member whose name or other unique identifier (such as address, in Desktop Manager objects) matches the string is returned. If *Identifier* is an integer, the member in that position in the collection object's list of members is returned. The integer option is usually available when the collection object has a Count, or similar property, from which the total number of members can be obtained.

Understanding Integer and String Identifiers

Not all collection objects accept an integer identifier, and not all accept a string identifier. The following table shows which objects accept which type of identifier.

Component	Object	Accepts Strings?	Accepts Integer?
Desktop Manager	Aliases	yes (addresses only)	no
	Groups	yes (")	no
	Servers	yes (")	no
	Workstations	yes (")	no
Queue Monitor	Jobs	yes*	yes
	Queues	yes	yes

* Accepts usernames only for the string identifier and returns the first job listed under that username.

Creating an Application

The following steps summarize how to create a Windows* application using the OLE API calls in this reference.

1. Create a project for the application.
2. Create the interface that enables you to interact with the application.
3. Enter the code in the code window.

   As a guide, and to save time, use portions of the code examples provided with the calls you are using.
4. Run and debug the code.
5. Create an executable (.EXE) file.
6. Add the executable file to your Windows desktop.

chapter 2

Desktop Manager OLE API

Desktop Manager is a ManageWise™ component that integrates real-time network diagnostics with inventory gathering and reporting. Desktop Manager helps you manage your network by enabling you to

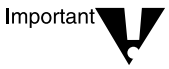
- Access workstations and servers in real time
- Manage device configurations, such as system configuration files
- Gather dynamic network and device information
- Perform network diagnostics
- Form complex inventory queries and save the results as an alias file

For more information about Desktop Manager, see *ManageWise 2.5 Desktop Management Guide*.

Launching Desktop Manager through OLE

The following statement launches Desktop Manager through OLE:

Set *ObjectVariable*=**CreateObject("DTM.Application")**



You must include this statement in any application that uses the Desktop Manager OLE API calls described in this chapter.

Object Hierarchy

The following hierarchy of OLE Automation objects indicates the structure by which you access the functionality of Desktop Manager:

```
Application object
  Workstations collection object
    Workstation object
  Servers collection object
    Server object
  Aliases collection object
    Alias object
  Groups collection object
    Group object
```

Object Attributes

You access specific features of Desktop Manager functionality by way of the objects and attributes (methods and properties) listed in Table 2-1.

Table 2-1
Object Attributes for Desktop Manager

Object	Method	Property
Application	InventoryQuery Quit ShowQueryDlg Find	Aliases FullName Groups InventoryServer Name Visible Workstations

Table 2-1 *continued*

Object Attributes for Desktop Manager

Object	Method	Property
Workstations collection		
Workstation	Copy	Address
	Cut	DHCPName
	DeviceInventory	IPAddress
	EditCfgFileDlg	IPXAddress
	FileTransfer	Name
	GetItemFromDatabase	TypeName
	GetItemFromUser	UserLoaded
	PrintCaptureQueue	
	Reboot	
	RunApplication	
	ShowMemoryMapDlg	
	ShowPingDlg	
	ShowServerPingDlg	
	ShowSummaryDlg	
	ViewWorkstation	
Servers collection		
Server	Attach	Address
	Child	Attached
	Copy	Children
	Cut	Name
	Detach	TypeName
		UserLoaded
Aliases collection		
Alias	Child	Children
	DeleteFile	Name
	Open	TypeName
	Paste	
	Save	
	SaveAs	
Groups collection		
Group	Child	Children
		Name
		TypeName

Application Object

The Application object is the root of Desktop Manager's object hierarchy. It contains methods and properties that perform inventory and non-device–specific functions. It also enables you to access the Workstations, Servers, Aliases, and Groups objects.

Aliases

Read-only property that returns the collection of all aliases on connected networks.

Syntax

ApplicationObject.**Aliases**

Return Value

Collection object from which individual alias group file objects can be obtained.

Example

```
Sub Main()
   Dim DTM as Object
   Dim Aliases As Object
   Dim A1 As Object

   Set DTM = CreateObject("DTM.Application")
   Set Aliases = DTM.Aliases
   Set A1 = Aliases("TEMP.STA")
   A1.Open("C:\WINDOWS\TEMP.STA")
End Sub
```

FullName

Read-only property that returns the full pathname of the application.

Syntax

ApplicationObject.**FullName**

Return Value

String containing the drive letter and the path where WSIGHT.EXE was launched.

Example

```
Sub Main()
   Dim DTM as Object
   Dim app$
   Set DTM = CreateObject("DTM.Application")
   app$ = DTM.FullName
End Sub
```

Groups

Read-only property that returns a collection of all query results, inventory databases, trees, and core data groups on connected networks.

Syntax

ApplicationObject.**Groups**

Return Value

Collection object from which individual group objects can be obtained.

Example

```
Sub Main()
   Dim DTM as Object
   Dim Groups As Object
   Dim G1 As Object

   Set DTM = CreateObject("DTM.Application")
   Set Groups = DTM.Groups
   Set G1 = Groups("Query Result")
End Sub
```

InventoryQuery

Method that queries all machines in the inventory database according to *FilterString* and returns the group object representing the query results.

Syntax

ApplicationObject.**InventoryQuery(***FilterString***)**

Parameters

FilterString

String.

Return Value

Group object.

Example

```
Sub Main()
   Dim DTM as Object
   Dim QResultGroup as Object

   Set DTM = CreateObject("DTM.Application")
   Set QResultGroup = DTM.InventoryQuery("Full Name =John Jones")
End Sub
```

InventoryServer

Read/write property that shows the name of the current inventory server for an application.

Syntax

ApplicationObject.**InventoryServer**

Parameters

None.

Return Value

Name of any server on the network.

Example

```
Sub Main()
   Dim DTM as Object
   Dim QResultGroup as Object

   Set DTM = CreateObject("DTM.Application")
   If DTM.InventoryServer <> "SERVER_NAME" Then
      DTM.InventoryServer = "SERVER_NAME"
   End If
End Sub
```

Name

Read-only property that returns the name of an application.

Syntax

ApplicationObject.**Name**

Return Value

Name of the application.

Example

```
Sub Main()
   Dim DTM as Object
   Dim appname$
   Set DTM = CreateObject("DTM.Application")
   appname$ = DTM.Name
End Sub
```

Quit

Method that terminates an application, no matter how many references to it exist.

Syntax

ApplicationObject.**Quit**

Parameters

None.

Return Value

None.

Example

```
Sub Main()
   Dim DTM as Object
   Set DTM = CreateObject("DTM.Application")
   DTM.Quit
End Sub
```

ShowQueryDlg

Method that brings up the Query Criteria dialog box in Desktop Manager. This method queries devices that have records in the currently selected database. The results are placed in a query results group on the device tree and returned as a Group object. BREQUEST must be loaded to use this method.

Syntax

ApplicationObject.**ShowQueryDlg**

Parameters

None.

Return Value

Group object representing the Query Results group. For more information, see “Group Object” on page 64.

Example

```
Sub Main()
   Dim DTM as Object
   Dim QResultGroup as Object

   Set DTM = CreateObject("DTM.Application")
   Set QResultGroup = DTM.ShowQueryDlg
End Sub
```

Visible

Standard read/write OLE property for all applications that sets or gets the visible state of the entire application. Setting the Visible property changes the state of active dialog boxes in an application. In ManageWise, the Desktop Manager comes up visible and minimized; however, you can use this property to hide it.

Syntax

ApplicationObject.**Visible(***Visible***)**

Parameters

Visible

Boolean: 0 = Not Visible, Nonzero = Visible.

Return Value

0 = Not Visible, Nonzero = Visible.

Example

```
Sub Main()
   Dim DTM as Object
   Set DTM = CreateObject("DTM.Application")
   DTM.Visible = 1
End Sub
```

Workstations

Read-only property that returns the collection of all workstations on connected networks.

Syntax

ApplicationObject.**Workstations**

Return Value

Collection object from which individual Workstation objects can be obtained.

Example

```
Sub Main()
   Dim DTM as Object
   Dim Wksts As Object
   Dim W1 As Object

   Set DTM = CreateObject("DTM.Application")
   Set Wksts = DTM.Workstations
   Set W1 = Wksts("11100111:0000AA123456")' assumes IPX
      or
   Set W1 = Wksts("IPX, 11100111:0000AA123456")
      or
   Set W1 = Wksts("IP,255.60.65.192")
      or
   Set W1 = Wksts("IP, USER1_MACHINE")
End Sub
```

Workstations Collection Object

The Workstations collection object is in the second level of Desktop Manager's object hierarchy. It contains the workstations that can be managed by Desktop Manager and enables you to access the Workstation object and its attributes. To access a single workstation within the collection, you specify the address.

Workstation Object

The Workstation object is in the third level of Desktop Manager's object hierarchy. It represents a single workstation discovered by Desktop Manager and provides methods and properties to manage it.

Address

Read-only property that gets the address of a workstation. The Address property for workstations is set by default. The address is the unique identifying string.

Syntax

WorkstationObject.**Address**

Return Value

String containing a workstation's address.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("00344300:00AB010FF311")
   addr$ = PC.Address
End Sub
```

Copy

Method that clears the clipboard and copies the workstation name or address to the clipboard. The workstation name or address can then be pasted into an Alias File object. If the application method Find was used, or if the object was obtained by calling the Child method from a Server, Alias, or Group object, the name copied to the clipboard is the login name. Otherwise, it is the network address with IMPORTED in parentheses.

Syntax

WorkstationObject.**Copy**

Parameters

None.

Return Value

None.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("00344300:00AB010FF311")
   PC.Copy
End Sub
```

Cut

Method that functions the same as the Copy method, but deletes the Workstation object from its parent group if the parent is an alias file. This is consistent with operations allowable on the device tree.

Syntax

WorkstationObject.**Cut**

Parameters

None.

Return Value

None.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("00344300:00AB010FF311")
   PC.Cut
End Sub
```

DeviceInventory

Method that shows an inventory of a workstation through the standard dialog box.

Syntax

WorkstationObject.**DeviceInventory**

Parameters

None.

Return Value

None.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("00344300:00AB010FF311")
   PC.DeviceInventory
End Sub
```

DHCPName

Read/write property that gets a workstation's Dynamic Host Configuration Protocol (DHCP) name. This property can be set by default by the Item method. In Desktop Manager, a Workstation object can have a different address/name for different supported protocols.

Syntax

WorkstationObject.**DHCPName**

Parameters

None.

Return Value

String containing a workstation's DHCP name.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("IP,USER1_MACHINE")
   addr$ = PC.DHCPName
End Sub
```

EditCfgFileDlg

Method that edits any database revision of a file on a remote workstation.

Syntax

WorkstationObject.**EditCfgFileDlg(***PropertyName***)**

Parameters

PropertyName

String containing the database attribute name that pulls up the appropriate file in the Edit dialog box.

Return Value

Nonzero = successful, 0 = unsuccessful. The return value is always 0 if an empty string is passed in.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("00344300:00AB010FF311")
   PC.EditCfgFileDlg("Cfg - c:\\windows\\win.ini")
End Sub
```

FileTransfer

Method that begins file transfer with a remote workstation.

Syntax

WorkstationObject.**FileTransfer**

Parameters

None.

Return Value

None.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("00344300:00AB010FF311")
   PC.FileTransfer
End Sub
```

GetItemFromDatabase

Method that gets the database value for the given item string.

Syntax

WorkstationObject.**GetItemFromDatabase(***Item***)**

Parameters

Item

String containing the name of the field item. For a list of possible field items, look at the Device Inventory for the device or print the device record. These fields are different from the user field names.

Return Value

String containing the value of the item selected. If the item is not in the database, **GetItemFromDatabase** returns an empty string.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("00344300:00AB010FF311")
   item$ = PC.GetItemFromDatabase("Processor - Speed")
End Sub
```

GetItemFromUser

Method that gets the value of the given item string from USERTSR.EXE.

Syntax

WorkstationObject.**GetItemFromUser(***Item***)**

Parameters

Item

String containing the name of the field item. These fields are different from the database field names.

Return Value

String containing the value of the item selected for the associated workstation if USERTSR.EXE (formerly USER.COM) responds on that workstation.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("00344300:00AB010FF311")
   item$ =PC.GetItemFromUser("CPUType")
End Sub
```

IPAddress

Read/write property that gets the IP address of a workstation. This property can be set by default by the Item method. In Desktop Manager, a Workstation object can have a different address/name for different supported protocols.

Syntax

WorkstationObject.**IPAddress**

Parameters

None.

Return Value

String containing a workstation's IP address.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("IP,10.60.65.192")
   addr$ = PC.IPAddress
End Sub
```

IPXAddress

Read/write property that gets the Internetwork Packet Exchange™ (IPX™) address of a workstation. This property can be set by default by the Item method. In Desktop Manager, a Workstation object can have a different address/name for the different supported protocols.

Syntax

WorkstationObject.**IPXAddress**

Parameters

None.

Return Value

String containing a workstation's IPX address.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("IPX,00344300:00AB010FF311")
   addr$ = PC.IPXAddress
End Sub
```

Name

Read/write property that gets or sets the name of a workstation. The Name property for workstations is not set automatically; Desktop Manager uses the address as the unique identifying string. This field must be set manually.

Syntax

WorkstationObject.**Name**

Parameters

None.

Return Value

String containing the name.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("00344300:00AB010FF311")
   PC.Name = "Joe's Computer"
End Sub
```

PrintCaptureQueue

Method that returns the status of the specified LPT port.

Syntax

WorkstationObject.**PrintCaptureQueue(***Lpt***)**

Parameters

Lpt

Integer between 1 and 3: LPT1 = 1, LPT2 = 2, LPT3 = 3.

Return Value

String containing one of the following values: “Not Captured” if the port is not captured to a network queue, or “SERVERNAME/QUEUE_NAME” if the port is captured to a network queue.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("00344300:00AB010FF311")
   capture$ = PC.PrintCaptureQueue(1)
End Sub
```

Reboot

Method that restarts a remote workstation.

Syntax

WorkstationObject.**Reboot**

Parameters

None.

Return Value

Boolean: nonzero if the machine was restarted, 0 if the machine was not restarted.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("00344300:00AB010FF311")
   PC.Reboot
End Sub
```

RunApplication

Method that runs an application on a remote workstation.

Syntax

WorkstationObject.**RunApplication(***AppName***)**

Parameters

AppName

String. If *AppName* is an empty string, a dialog box prompts for a remote program name. Otherwise, the program string can contain any application name (with a path) and parameters separated by spaces.

Return Value

Boolean: nonzero = successful, 0 = it failed. The return value is always 0 if an empty string is passed in.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("00344300:00AB010FF311")
   PC.RunApplication"C:\WINDOWS\WRITE.EXE TEMP.TXT"
End Sub
```

ShowMemoryMapDlg

Method that displays the Memory Map dialog box for the workstation.

Syntax

WorkstationObject.**ShowMemoryMapDlg**

Parameters

None.

Return Value

None.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("00344300:00AB010FF311")
   PC.ShowMemoryMapDlg
End Sub
```

ShowPingDlg

Method that displays the Workstation Ping Test dialog box.

Syntax

WorkstationObject.**ShowPingDlg**

Parameters

None.

Return Value

None.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("00344300:00AB010FF311")
   PC.ShowPingDlg
End Sub
```

ShowServerPingDlg

Method that displays the Server Ping Test dialog box.

Syntax

WorkstationObject.**ShowServerPingDlg**

Parameters

None.

Return Value

None.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("00344300:00AB010FF311")
   PC.ShowServerPingDlg
End Sub
```

ShowSummaryDlg

Method that displays the Workstation Summary dialog box.

Syntax

WorkstationObject.**ShowSummaryDlg**

Parameters

None.

Return Value

None.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("00344300:00AB010FF311")
   PC.ShowSummaryDlg
End Sub
```

TypeName

Read-only property that returns the type of the current object, which can be Workstation, Server, Alias, or Group.

Syntax

WorkstationObject.**TypeName**

Return Value

String containing one of the following values: Workstation, Server, Alias, or Group.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("00344300:00AB010FF311")
   typename$ = PC.TypeName
End Sub
```

UserLoaded

Read-only property that returns a nonzero Boolean value (“true”) if the User agent is loaded on the given workstation.

Syntax

WorkstationObject.**UserLoaded**

Return Value

Boolean: nonzero = the User agent is loaded, 0 = the User agent is not loaded.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("00344300:00AB010FF311")
   bUserLoaded% = PC.UserLoaded
End Sub
```

ViewWorkstation

Method that enables you to control a remote workstation from the ManageWise Console.

Syntax

WorkstationObject.**ViewWorkstation**

Parameters

None.

Return Value

None.

Example

```
Sub Main()
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set PC = DTM.Workstations("00344300:00AB010FF311")
   PC.ViewWorkstation
End Sub
```

Servers Collection Object

The Servers collection object is in the second level of Desktop Manager's object hierarchy. It contains the servers that can be managed by Desktop Manager and enables you to access the Server object and its attributes.

Server Object

The Server object is in the third level of Desktop Manager's object hierarchy. It represents a single server discovered by Desktop Manager and provides methods and properties to manage it.

Address

Read/write property that gets the address of a server. The Address property for servers is not set by default. The name is the unique identifying string for such objects. The Address property must be set manually.

Syntax

ServerObject.**Address**

Return Value

String containing a server's address.

Example

```
Sub Main()
   Dim Server as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")

   Set Server = DTM.Servers("SERVER_NAME")
   addr$ = Server.Address
End Sub
```

Attach

Method that attaches a NetWare® workstation to the server of the given name. If the login name or the password fields are empty strings, or if the attach fails, the Attach dialog box appears.

Syntax

ServerObject.**Attach(***loginName*, *password***)**

Parameters

loginName

String containing a valid login name on the server.

password

String containing the password associated with the login name.

Return Value

Boolean: nonzero = attach was successful, 0 = attach failed.

Example

```
Sub Main()
   Dim Server as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")

   Set Server = DTM.Servers("SERVER_NAME")
   retVal% = Server.Attach( "", "")
   retVal% = Server.Attach("User1", "password")
End Sub
```

Attached

Read-only property that returns a nonzero Boolean value (“true”) if the ManageWise Console is attached to the specified server.

Syntax

ServerObject.**Attached**

Return Value

Boolean: nonzero = attached, 0 = not attached.

Example

```
Sub Main()
   Dim Server as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")

   Set Server = DTM.Servers("SERVER_NAME")
   bAttached% = Server.Attached
End Sub
```

Child

Method that returns either the 0-based nth child (if an integer is passed in) or the child that matches the variable string. The string is compared against the full address of the child workstation and the login name.

Syntax

ServerObject.**Child(***var***)**

Parameters

var

Variable type that passes an integer or a string. As an integer, it contains the index of the child to be retrieved. As a string, it contains the name or address of the child to be retrieved.

Return Value

Workstation object.

Example

```
Sub Main()
   Dim Server as Object
   Dim child as Object
   Dim DTM as Object
   Set DTM = CreateObject("DTM.Application")

   Set Server = DTM.Servers("SERVER_NAME")
   Set child = Server.Child("00344300:00AB010FF312")
   Set child = Server.Child(3)
End Sub
```

Children

Read-only property that returns the number of children of the Server object. This represents the number of logged-in users for a given server.

Syntax

ServerObject.**Children**

Return Value

Integer representing the number of workstations currently logged in to the server.

Example

```
Sub Main()
   Dim Server as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")

   Set Server = DTM.Servers("SERVER_NAME")
   nCount% = Server.Children
End Sub
```

Copy

Method that clears the clipboard and copies the server to the clipboard. The server is pasted into an Alias File object. If the application method Find was used, or if the object was obtained by calling the Child method from a Group or Alias object, the name copied to the clipboard is the server name. Otherwise, it is the name with IMPORTED in parentheses.

Syntax

ServerObject.**Copy**

Parameters

None.

Return Value

None.

Example

```
Sub Main()
   Dim Server as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")

   Set Server = DTM.Servers("SERVER_NAME")
   Server.Copy
End Sub
```

Cut

Method that works the same as the Copy method, but deletes the Server object from its parent group only if the parent is an alias file. This is consistent with operations allowable in the logical Network View window.

Syntax

ServerObject.**Cut**

Parameters

None.

Return Value

None.

Example

```
Sub Main()
   Dim Server as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")

   Set Server = DTM.Servers("SERVER_NAME")
   Server.Cut
End Sub
```

Detach

Method that detaches the ManageWise Console from a server.

Syntax

ServerObject.**Detach**

Parameters

None.

Return Value

None.

Example

```
Sub Main()
   Dim Server as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")

   Set Server = DTM.Servers("SERVER_NAME")
   Server.Detach
End Sub
```

Name

Read/write property that enables you to view or change the alias of a selected server. The Name property for servers is set to the actual name of the server by default. The name is the unique identifying string.

Syntax

ServerObject.**Name**

Parameters

None.

Return Value

String containing the name of the server.

Example

```
Sub Main()
   Dim Server as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set Server = DTM.Servers("SERVER_NAME")
   Server.Name = "NEW_NAME"
End Sub
```

TypeName

Read-only property that returns the type of the current object (Workstation, Server, Alias, or Group).

Syntax

ServerObject.**TypeName**

Return Value

String containing one of the following values: Workstation, Server, Alias, or Group.

Example

```
Sub Main()
   Dim Server as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")

   Set Server = DTM.Servers("SERVER_NAME")
   typename$ = Server.TypeName
End Sub
```

UserLoaded

Read-only property that returns a nonzero Boolean value (“true”) if USER.NLM is loaded on the given server.

Syntax

ServerObject.**UserLoaded**

Return Value

Boolean: nonzero = the User agent is loaded, 0 = the User agent is not loaded.

Example

```
Sub Main()
   Dim Server as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")

   Set Server = DTM.Servers("SERVER_NAME")
   bUserLoaded = Server.UserLoaded
End Sub
```

Aliases Collection Object

The Aliases collection object is the second level of Desktop Manager's object hierarchy. It contains user-defined aliases and enables you to access the Alias object and its attributes.

Alias Object

The Alias object is the third level of Desktop Manager's object hierarchy. It represents a single user-defined alias and provides methods and properties to manage it.

Child

Method that returns either the 0-based nth child (if an integer is passed in) or the child that matches the variable string. If the child is a workstation or server, the string is compared against the full address and the name of the child (returning whichever matches); otherwise, the string is compared with the child's name.

Syntax

AliasObject.**Child(***var***)**

Parameters

var

Variable type that passes an integer or a string. As an integer, it contains the index of the child to be retrieved. As a string, it contains the name or address of the child to be retrieved.

Return Value

Workstation, Server, or Group object.

Example

```
Sub Main()
   Dim AliasFile as Object
   Dim child as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")

   Set AliasFile = DTM.Aliases("TEMP.STA")
   Set child = AliasFile.Child("BSMITH")
   Set child = AliasFile.Child("00344300:00AB010FF312")
   Set child = AliasFile.Child(3)
   type$ = child.TypeName
End Sub
```

Children

Read-only property that returns the number of children of the Alias File object.

Syntax

AliasObject.**Children**

Return Value

Integer representing the number of children under an alias.

Example

```
Sub Main()
   Dim AliasFile as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")

   Set AliasFile = DTM.Aliases("temp.sta")
   nCount% = AliasFile.Children
End Sub
```

DeleteFile

Method that deletes the alias file from the disk and from the Network View window. No other alias operations except Open can be performed after this.

Syntax

AliasObject.**DeleteFile**

Parameters

None.

Return Value

None.

Example

```
Sub Main()
   Dim AliasFile as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")

   Set AliasFile = DTM.Aliases("temp.sta")
   AliasFile.Open "c:\windows\temp.sta"
   AliasFile.DeleteFile
End Sub
```

Name

Read-only property that gets the name of an alias. The Name property for aliases is set by default. The name is the unique identifying string for the file, although there might be many instances of the alias file in the Network View window.

Syntax

AliasObject.**Name**

Return Value

String containing the name of the alias set.

Example

```
Sub Main()
   Dim AliasFile as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")

   Set AliasFile = DTM.Aliases("temp.sta")
   filename$ = AliasFile.Name
End Sub
```

Open

Method that opens an existing alias file into the current Alias object in the Network View window. The name must contain the full path.

Syntax

AliasObject.**Open(***Name***)**

Parameters

Name

String containing the full path and name of the alias file to be opened.

Return Value

Boolean: nonzero if successful, 0 if unsuccessful.

Example

```
Sub Main()
   Dim AliasFile as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")
   Set AliasFile = DTM.Aliases("temp.sta")
   AliasFile.Open ("c:\windows\temp.sta")
End Sub
```

Paste

Method that pastes elements on the clipboard to the Alias File object.

Syntax

AliasObject.**Paste**

Parameters

None.

Return Value

None.

Example

```
Sub Main()
   Dim AliasFile as Object
   Dim PC as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")

   Set AliasFile = DTM.Aliases("temp.sta")
   AliasFile.Open "c:\windows\temp.sta"

   PC = DTM.Find("BSMITH")
   PC.Copy
   AliasFile.Paste
End Sub
```

Save

Method that saves the Alias File object to the path specified either through Open or a previous SaveAs. If no path was previously specified, the Save As dialog box opens for the user to select a path and filename.

Syntax

AliasObject.**Save**

Parameters

None.

Return Value

Boolean: nonzero if successful, 0 if it failed.

Example

```
Sub Main()
   Dim AliasFile as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")

   Set AliasFile = DTM.Aliases("temp.sta")
   AliasFile.Open "c:\windows\temp.sta"
   AliasFile.Save
End Sub
```

SaveAs

Method that saves the Alias File object to *Name*. If no path is previously specified, the Save As dialog box opens for the user to select a path and filename.

Syntax

AliasObject.**SaveAs(***Name***)**

Parameters

Name

String containing the full path and name to where the alias file will be saved.

Return Value

Boolean: nonzero if it was successful, 0 if it failed.

Example

```
Sub Main()
   Dim AliasFile as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")

   Set AliasFile = DTM.Aliases("temp.sta")
   AliasFile.Open "c:\windows\temp.sta"
   AliasFile.SaveAs("f:\temp.sta")
End Sub
```

TypeName

Read-only property that returns the type of the current object (Workstation, Server, Alias, or Group).

Syntax

AliasObject.**TypeName**

Return Value

String containing one of the following values: Workstation, Server, Alias, or Group.

Example

```
Sub Main()
   Dim AliasFile as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")

   Set AliasFile = DTM.Find("TEMP.STA")
   typename$ = AliasFile.TypeName
End Sub
```

Groups Collection Object

The Groups collection object is the second level of Desktop Manager's object hierarchy. It contains all groups in the Network View window other than aliases, and enables you to manage devices relative to where they are in the Network View window.

Group Object

The Group object is the third level of Desktop Manager's object hierarchy. It represents a single group in the Network View window and provides methods and properties to manage it.

Child

Method that returns either the 0-based nth child (if an integer is passed in) or the child that matches the variable string. If the child is a workstation or server, the string is compared against the full address and the name of the child (returning whichever matches); otherwise, the string is compared with the child's name.

Syntax

GroupObject.**Child(***var***)**

Parameters

var

Variable type that passes an integer or a string. As an integer, it contains the index of the child to be retrieved. As a string, it contains the name or address of the child to be retrieved.

Return Value

Workstation, Server, or Group object.

Example

```
Sub Main()
   Dim Group as Object
   Dim child as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")

   Set Group = DTM.Groups("Query Result")
   Set child = Group.Child("BSMITH")
   Set child = Group.Child("00344300:00AB010FF312")
   Set child = Group.Child(3)
End Sub
```

Children

Read-only property that returns the number of children of the Group object. This represents the number of children under a particular group.

Syntax

GroupObject.**Children**

Return Value

Integer representing the number of children under a group.

Example

```
Sub Main()
   Dim Group as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")

   Set Group = DTM.Groups("Query Result")
   nCount% = Group.Children
End Sub
```

Name

Read/write property that gets or sets the name of a group. The Name property for groups is set by default. The name is the unique identifying string.

Syntax

GroupObject.**Name**

Parameters

None.

Return Value

String containing the group name.

Example

```
Sub Main()
   Dim Group as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")

   Set Group = DTM.Groups("Query Result")
   ‘read the group name
   groupname$ = Group.Name
   ‘change (write) the group name
   Group.Name = "QR1"
End Sub
```

TypeName

Read-only property that returns the type of the current object (Workstation, Server, Alias, or Group).

Syntax

GroupObject.**TypeName**

Return Value

String containing one of the following values: Workstation, Server, Alias, or Group.

Example

```
Sub Main()
   Dim Group as Object
   Dim DTM as Object

   Set DTM = CreateObject("DTM.Application")

   Set Group = DTM.Find("Attached Servers")
   name$ = Group.TypeName
End Sub
```

chapter 3

Queue Monitor OLE API

Queue Monitor is a ManageWise™ component that enables you to control network print queues. You use Queue Monitor to

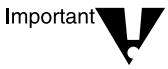
- Monitor print queues and print jobs
- Manage network print queues
- Manipulate queued print jobs

For more information about Queue Monitor, see *ManageWise 2.5 Desktop Management Guide.*

Launching Queue Monitor through OLE

The following statement launches Queue Monitor through OLE:

Set *ObjectVariable*=**CreateObject("QMon.Application")**



You must include this statement in any application that uses the Queue Monitor OLE API calls described in this chapter.

Object Hierarchy

The following hierarchy of OLE Automation objects indicates the structure by which you access the functionality of Queue Monitor:

```
Application object
  Queues collection object
    Queue object
    Jobs collection object
    Job object
```

Object Attributes

You access specific features of Queue Monitor functionality by way of the objects and attributes (methods and properties) listed in Table 3-1.

Table 3-1
Object Attributes for Queue Monitor

Object	Method	Property
Application	Quit	Queues Visible
Queues collection	Add Remove Select SelectAll	ByteCount Count JobCount
Queue		ByteCount Jobs Name ServerName Status
Jobs collection	CopyAll DeleteAll DeleteJob HoldAll MoveAll	Count

Table 3-1 *continued*
Object Attributes for Queue Monitor

Object	Method	Property
Job	CopyJob MoveJob ShowStatus	ByteCount ID JobName OperatorHold Owner UserHold

Application Object

The Application object is the root of Queue Monitor's object hierarchy. It possesses attributes that control the appearance and function of Queue Monitor. It also enables you to access the Queues object.

Queues

Read-only property that returns an empty collection of Queue objects. Queue objects can be added and managed.

Syntax

ApplicationObject.**Queues**

Return Value

Empty collection object.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim Queues As Object
   Dim rv As Boolean

   Set Qmon = CreateObject("QMon.Application")
   Set Queues = Qmon.Queues
   ' Add a new queue to monitor to the collection of queues.
   rv = Queues.Add("BRANDX/IIISI_QUEUE")
End Sub
```

Quit

Method that terminates an application no matter how many references to the application exist.

Syntax

ApplicationObject.**Quit**

Parameters

None.

Example

```
Sub Main()
   Dim Qmon As Object

   Set Qmon = CreateObject("QMon.Application")
   ' Show the application window
   Qmon.Visible = True
   ' Now quit Qmonitor
   Qmon.Quit
End Sub
```

Visible

Read/write property that gets or sets the visibility of the application window.

Syntax

ApplicationObject.**Visible**

Parameters

None.

Remarks

The following values are returned when reading the Visible property and are used to set the Visible property:

True = Visible.

False = Not visible.

Return Value

Boolean containing the visibility status.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim bVisible As Boolean

   Set Qmon = CreateObject("QMon.Application")
   'Show the application window
   Qmon.Visible = True
   'Now get the visibility status
   bVisible = Qmon.Visible
   'Now quit Qmonitor
   Qmon.Quit
End Sub
```

Queues Collection Object

The Queues collection object is the second level of Queue Monitor's object hierarchy. It contains a list of all Queue objects currently being monitored by the component, as well as attributes to manage queues as a collection.

Add

Method that inserts the specified queue into the collection of queues being monitored.

Syntax

QueuesObject.**Add(***QueueName***)**

Parameters

QueueName

A string that represents the name of the queue you want to add to the Queue Monitor window. The string should have the following format:

ServerName/*QueueName*

Return Value

Boolean identifying the success of the method: True = success, False = failure.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim Queues As Object
   Dim rv As Boolean

   Set Qmon = CreateObject("QMon.Application")
   Set Queues = Qmon.Queues
   'Add a new queue to monitor to the collection of queues.
   rv = Queues.Add("BRANDX/IIISI_QUEUE")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   end if
End Sub
```

ByteCount

Read-only property that returns the number of bytes contained in all print jobs within all print queues displayed in the Queue Monitor window. This property is updated as individual print jobs are added and serviced.

Syntax

QueuesObject.**ByteCount**

Return Value

Long integer.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim Queues As Object
   Dim rv As Boolean
   Dim bytes As Long

   Set Qmon = CreateObject("QMon.Application")
   Set Queues = Qmon.Queues
   'Add a new queue to monitor to the collection of queues.
   rv = Queues.Add("BRANDX/IIISI_QUEUE")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   else
      'Get the total number of bytes in jobs in
      'queues
      bytes = Queues.ByteCount
   end if
End Sub
```

Count

Read-only property that returns the number of print queues currently displayed in the Queue Monitor window.

Syntax

QueuesObject.**Count**

Return Value

Long integer.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim Queues As Object
   Dim rv As Bool
   Dim qCount As Long

   Set Qmon = CreateObject("QMon.Application")
   Set Queues = Qmon.Queues
   ' Add a new queue to the collection of queues
   rv = Queues.Add("BRANDX/IIISI_QUEUE")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   else
      ' Get the total of queues being monitored
      qCount = Queues.Count
   end if
End Sub
```

JobCount

Read-only property that returns the number of jobs contained in all print queues currently displayed in the Queue Monitor window. This property is updated as individual print jobs are added and serviced.

Syntax

QueuesObject.**JobCount**

Return Value

Long integer.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim Queues As Object
   Dim rv As Boolean
   Dim jCount As Long

   Set Qmon = CreateObject("QMon.Application")
   Set Queues = Qmon.Queues
   ' Add a new queue to monitor to the collection of queues.
   rv = Queues.Add("BRANDX/IIISI_QUEUE")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   else
      ' Get the total number of jobs in queues
      jCount = Queues.JobCount
   end if
End Sub
```

Remove

Method that removes a print queue from the Queue Monitor window and stops tracking its activity.

Syntax

QueuesObject.**Remove** *Value*

Parameters

Value

Variable type that passes an integer or a string. As an integer, it contains the index of the queue within the queue collection. As a string, it contains the unique name of the Queue object, which is removed from the Queue Monitor window.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim Queues As Object
   Dim rv As Boolean

   Set Qmon = CreateObject("QMon.Application")
   Set Queues = Qmon.Queues
   'Add a new queue to the collection of queues
   rv = Queues.Add("BRANDX/IIISI_QUEUE1")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   end if
   rv = Queues.Add("BRANDX/IIISI_QUEUE2")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   end if

   'Remove the queues that we've added

   ' This removes IIISI_QUEUE1 because it was the
   'first queue added to the collection.
   Queues.Remove 0
   ' This removes IIISI_QUEUE2 by its unique name
   Queues.Remove "BRANDX/IIISI_QUEUE2"
End Sub
```

Select

Method that displays Queue Monitor's Select Queues dialog box. The user can select queues to be displayed by Queue Monitor. When the user clicks OK, the selected Queues automatically replace any queues previously displayed in the Queue Monitor window.

Syntax

QueuesObject.**Select**

Parameters

None.

Return Value

None.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim Queues As Object

   Set Qmon = CreateObject("QMon.Application")
   Set Queues = Qmon.Queues
   Queues.Select
End Sub
```

SelectAll

Method that fills the Queue Monitor window with all the queues that Queue Monitor can discover on all the attached servers.

Syntax

QueuesObject.**SelectAll**

Parameters

None.

Return Value

None.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim Queues As Object

   Set Qmon = CreateObject("QMon.Application")
   Set Queues = Qmon.Queues
   'Select all available queues
   Queues.SelectAll
End Sub
```

Queue Object

The Queue object is the third level of Queue Monitor's object hierarchy. It represents a single print queue displayed in Queue Monitor and provides the methods and properties to manage that print queue. The Queue object also contains a Jobs collection object, which provides access to each job within the queue.

ByteCount

Read-only property that returns the number of bytes contained in all print jobs within a given print queue. This property is updated as individual print jobs are added and serviced.

Syntax

QueueObject.**ByteCount**

Return Value

Long integer.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim Queues As Object
   Dim bytes As Long

   Set Qmon = CreateObject("QMon.Application")
   Set Queues = Qmon.Queues
   ' Add a new queue to those being monitored.
   rv = Queues.Add("BRANDX/IIISI_QUEUE")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   else
      'Get the total number of bytes used by jobs
      'in the queue just added.
      bytes = Queues(0).ByteCount
   end if
End Sub
```

Jobs

Read-only property that gets a collection of jobs in a given queue.

Syntax

QueueObject.**Jobs**

Return Value

Reference to a Jobs collection object, or NULL if there are no jobs in the queue.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim Queues As Object
   Dim Jobs As Object
   Dim rv As Boolean

   Set Qmon = CreateObject("QMon.Application")
   Set Queues = Qmon.Queues
   'Add a new queue to the collection of queues.
   rv = Queues.Add("BRANDX/IIISI_QUEUE")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   else
      'Get the collection of jobs being serviced by
      'the queue.
      Set Jobs = Queues(0).Jobs
   end if
End Sub
```

Name

Read-only property that gets the name of the specified print queue. This name does not include the name of the server on which the print queue resides.

Syntax

QueueObject.**Name**

Return Value

String.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim Queues As Object
   Dim qCount As Long
   Dim i As Long
   Dim qName As String

   Set Qmon = CreateObject("QMon.Application")
   Set Queues = Qmon.Queues
   'Select all queues on attached servers.
   Queues.SelectAll
   qCount = Queues.Count
   for i = 0 To qCount - 1
      qName = Queues(i).Name
      'Print the name of each queue.
      MsgBox("Queue name = " + qName)
   next i
End Sub
```

ServerName

Read-only property that gets the name of the server on which a print queue resides.

Syntax

QueueObject.**ServerName**

Return Value

String.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim Queues As Object
   Dim qCount As Long
   Dim i As Long
   Dim qName As String
   Dim sName As String

   Set Qmon = CreateObject("QMon.Application")
   Set Queues = Qmon.Queues
   ' Select all queues on attached servers.
   Queues.SelectAll
   qCount = Queues.Count
   for i = 0 To qCount - 1
      qName = Queues(i).Name
      sName = Queues(i).ServerName
      ' Print the queue name and server name.
      MsgBox("Full Queue name = " + sName + "/" + _
      qName)
   next i
End Sub
```

Status

Read-only property that gets the current status of a print queue.

Syntax

QueueObject.**Status**

Return Value

Long number containing the queue status, which can be any combination of the following:

1 = User cannot add jobs to the queue.

2 = New print server cannot be attached to the queue.

4 or 9 = Attached server cannot service the queue.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim qStatus As Long

   Set Qmon = CreateObject("QMon.Application")

   ' Add a new queue to the collection of queues.
   rv = Qmon.Queues.Add("BRANDX/IIISI_QUEUE1")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   else
      qStatus = Qmon.Queues(0).Status
      if (qStatus = 1) Then
         MsgBox("User cannot add jobs to queue.")
      else if (qStatus = 5)
         MsgBox("Attached printer server cannot " _
      + "service jobs, and user can't add " _
            + "jobs to queue.")
      end if
   end if
End Sub
```

Jobs Collection Object

The Jobs collection object contains a list of all Job objects within a given Queue object. The Jobs collection object also contains methods and properties to manage jobs as a group within a print queue.

CopyAll

Method that copies all print jobs from a selected queue to another queue specified by *QueueName*.

Syntax

JobsObject.**CopyAll** *QueueName*

Parameters

QueueName

String containing the name of the queue to which you want to copy the print jobs. The string should have the following format:

ServerName/QueueName

Return Value

None.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim Queues As Object
   Dim rv As Boolean
   Set Qmon = CreateObject("QMon.Application")
   Set Queues = Qmon.Queues
   ' Add a new queue to the collection of queues.
   rv = Queues.Add("BRANDX/IIISI_QUEUE1")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   end if
   rv = Queues.Add("BRANDX/IIISI_QUEUE2")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   end if

   'Move all jobs from IIISI_QUEUE1 to IIISI_QUEUE2
   Queues(0).Jobs.CopyAll "BRANDX/IIISI_QUEUE2"
End Sub
```

Count

Read-only property that gets the number of print jobs in a given queue.

Syntax

JobsObject.**Count**

Return Value

Long number containing the number of jobs contained in a given print queue.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim jCount As Object

   Set Qmon = CreateObject("QMon.Application")
   'Add a new queue to the collection of queues.
   rv = Qmon.Queues.Add("BRANDX/IIISI_QUEUE1")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   end if

   'Gets the number of jobs serviced by the queue
   jCount = Qmon.Queues(0).Jobs.Count
End Sub
```

DeleteAll

Method that deletes all print jobs from a given print queue.

Syntax

JobsObject.**DeleteAll**

Parameters

None.

Return Value

None.

Example

```
Sub Main()
   Dim Qmon As Object

   Set Qmon = CreateObject("QMon.Application")
   ' Add a new queue to monitor to the collection of queues.
   rv = Qmon.Queues.Add("BRANDX/IIISI_QUEUE1")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   end if

   ' Delete all the print jobs in the first queue being monitored.
   Qmon.Queues(0).Jobs.DeleteAll
End Sub
```

DeleteJob

Method that deletes a specific job from a print queue.

Syntax

JobsObject.**DeleteJob** *Value*

Parameters

Value

String containing the unique name of the Job object, or integer containing the index of the job within the Jobs collection. The Job object is deleted from the specified queue.

Return Value

None.

Example

```
Sub Main()
   Dim Qmon As Object

   Set Qmon = CreateObject("QMon.Application")
   ' Add a new queue to the collection of queues.
   rv = Qmon.Queues.Add("BRANDX/IIISI_QUEUE1")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   end if

   ' Delete the first queue.
   Qmon.Queues(0).Jobs.DeleteJob 0
End Sub
```

HoldAll

Method that puts user or operator hold on all print jobs in a given queue.

Syntax

JobsObject.**HoldAll** *HoldType*

Parameters

HoldType

Integer containing the type of hold being placed on print jobs:

1 = user hold

2 = operator hold

Return Value

None.

Example

```
Sub Main()
   Dim Qmon As Object

   Set Qmon = CreateObject("QMon.Application")
   ' Add a new queue to the collection of queues.
   rv = Qmon.Queues.Add("BRANDX/IIISI_QUEUE1")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   end if

   ' Place a User Hold on all print jobs.
   Qmon.Queues(0).Jobs.HoldAll 1
End Sub
```

MoveAll

Method that moves all print jobs from a selected queue to another queue specified by *QueueName*. The specified queue does not need to be visible in the Queue Monitor windows, but you must be attached to the server on which the queue resides.

Syntax

JobsObject.**MoveAll** *QueueName*

Parameters

QueueName

String containing the name of the queue to which you want to move the print jobs. The string should have the following format:

ServerName / *QueueName*

Return Value

None.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim rv As Boolean

   Set Qmon = CreateObject("QMon.Application")
   ' Add a new queue to the collection of queues.
   rv = Qmon.Queues.Add("BRANDX/IIISI_QUEUE1")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   end if
   rv = Qmon.Queues.Add("BRANDX/IIISI_QUEUE2")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   end if
   'Move all jobs from IIISI_QUEUE1 to IIISI_QUEUE2
   Qmon.Queues(0).Jobs.MoveAll "BRANDX/IIISI_QUEUE2"
End Sub
```

Job Object

The Job object represents a single print job and contains the methods and properties to manage single print jobs.

When obtaining a Job object from the Jobs collection object, a string containing a username can be given instead of an ordinal. In this case, Queue Monitor looks for the first job in the queue listed under the username and returns that object. If multiple jobs are present under the same username, an ordinal must be used to obtain objects of the other jobs.

ByteCount

Read-only property that returns the size of the print job in bytes. This property is updated as the print job is added to the print queue.

Syntax

JobObject.**ByteCount**

Return Value

Long number containing the size of the print job in bytes.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim rv As Boolean
   Dim jSize As Long

   Set Qmon = CreateObject("QMon.Application")
   ' Add a new queue to the collection of queues.
   rv = Qmon.Queues.Add("BRANDX/IIISI_QUEUE1")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   else
      ' Now get the size of the first job in the queue
      jSize = Qmon.Queues(0).Jobs(0).ByteCount
   end if
End Sub
```

CopyJob

Method that copies the job to the specified queue. The specified queue does not need to be visible in the Queue Monitor window, but you must be attached to the server on which the specified queue resides.

Syntax

JobObject.**CopyJob** *QueueName*

Parameters

QueueName

String that contains the name of the queue to which you want the job copied. The string should have the following format:

ServerName/*QueueName*

Return Value

None.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim rv As Boolean

   Set Qmon = CreateObject("QMon.Application")
   ' Add a new queue to the collection of queues.
   rv = Qmon.Queues.Add("BRANDX/IIISI_QUEUE1")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   end if

   ' Copy the job to IIISI_QUEUE2
   Qmon.Queues(0).Jobs(0).CopyJob "BRANDX/IIISI_QUEUE2"
End Sub
```

ID

Read-only property that gets the job ID assigned to the job by the printing service.

Syntax

JobObject.**ID**

Return Value

Long number containing the job ID.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim id As Long
   Dim rv As Boolean

   Set Qmon = CreateObject("QMon.Application")
   ' Add a new queue to the collection of queues.
   rv = Qmon.Queues.Add("BRANDX/IIISI_QUEUE1")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   end if

   ' Get the ID for the job
   id = Qmon.Queues(0).Jobs(0).ID
End Sub
```

JobName

Read-only property that gets the job name assigned to a job by the printing service.

Syntax

JobObject.**JobName**

Return Value

String containing the job name.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim jName As String
   Dim rv As Boolean

   Set Qmon = CreateObject("QMon.Application")
   ' Add a new queue to the collection of queues.
   rv = Qmon.Queues.Add("BRANDX/IIISI_QUEUE1")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   end if

   ' Get the ID for this job
   jName = Qmon.Queues(0).Jobs(0).JobName
End Sub
```

MoveJob

Method that moves the job to the specified queue. The queue does not need to be visible in the Queue Monitor window, but you must be attached to the server on which the queue resides.

Syntax

JobObject.**CopyJob** *QueueName*

Parameters

QueueName

String containing the name of the queue to which you want to move the job. The string should have the following format:

ServerName / *QueueName*

Parameters

None.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim rv As Boolean

   Set Qmon = CreateObject("QMon.Application")
   ' Add a new queue to the collection of queues.
   rv = Qmon.Queues.Add("BRANDX/IIISI_QUEUE1")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   end if

   ' Now move this job to IIISI_QUEUE2
   Qmon.Queues(0).Jobs(0).MoveJob "BRANDX/IIISI_QUEUE2"
End Sub
```

OperatorHold

Read/write property that gets or sets the operator hold status of a print job.

Syntax

JobObject.**OperatorHold**

Remarks

Use the following values to change the hold status:

1 = Places the job on operator hold.

0 = Removes the operator hold from the job.

Return Value

Boolean: 1 = job is being held by the operator, 0 = job is not being held by the operator.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim rv As Bool

   Set Qmon = CreateObject("QMon.Application")
   ' Add a new queue to the collection of queues.
   rv = Qmon.Queues.Add("BRANDX/IIISI_QUEUE1")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   end if

   ' Place an operator hold on this job.
   Qmon.Queues(0).Jobs(0).OperatorHold = 1
   ' Get the hold status
   rv = Qmon.Queues(0).Jobs(0).OperatorHold
End Sub
```

Owner

Read-only property that gets the job owner name assigned to the job by the printing service.

Syntax

JobObject.**Owner**

Return Value

String containing the job owner name.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim oName As String

   Set Qmon = CreateObject("QMon.Application")
   ' Add a new queue to the collection of queues.
   rv = Qmon.Queues.Add("BRANDX/IIISI_QUEUE1")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   end if

   ' Get the owner Name for the job
   oName = Qmon.Queues(0).Jobs(0).Owner
End Sub
```

ShowStatus

Method that displays the Queue Monitor Job Status dialog box for a given job.

Syntax

JobObject.**ShowStatus**

Parameters

None.

Return Value

None.

Example

```
Sub Main()
   Dim Qmon As Object

   Set Qmon = CreateObject("QMon.Application")
   ' Add a new queue to the collection of queues.
   rv = Qmon.Queues.Add("BRANDX/IIISI_QUEUE1")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   end if

   ' Show the Status dialog box for this job
   Qmon.Queues(0).Jobs(0).ShowStatus
End Sub
```

UserHold

Read/write property that gets or sets the user hold status of the print job.

Syntax

JobObject.**UserHold**

Remarks

Use the following values to change the hold status:

1 = Place the job on user hold.

0 = Remove the user hold from the job.

Return Value

Boolean: 1= job is being held by the user, 0 = job is not being held by the user.

Example

```
Sub Main()
   Dim Qmon As Object
   Dim rv As Boolean

   Set Qmon = CreateObject("QMon.Application")
   ' Add a new queue to the collection of queues.
   rv = Qmon.Queues.Add("BRANDX/IIISI_QUEUE1")
   if (rv = False) Then
      MsgBox("Unable to add queue")
   end if

   ' Place a user hold on this job.
   Qmon.Queues(0).Jobs(0).UserHold = 1
   ' Get the hold status
   rv = Qmon.Queues(0).Jobs(0).UserHold
End Sub
```