NetIQ OLE Object Reference Guide

NetIQ AppManager

Version 6.0

Legal Notice

Chapter 1

NetIQOLE object reference

Disclaimer

NetIQ provides the NetIQOLE software and related information "as is" without warranties of any kind, expressed or implied. Under no circumstances will NetIQ be liable for any loss of or damage to your data, lost profits, lost savings, third-party damages, incidental damages or other economic consequential damages. Usage of software and related information is subject to the terms of the NetIQ AppManager Software License Agreement. This material is subject to change at any time and interfaces are not guaranteed to be preserved or compatible with future releases.

This chapter covers the NetIQOLE OLE automation object, and the objects to which it provides access. The following topics are covered:

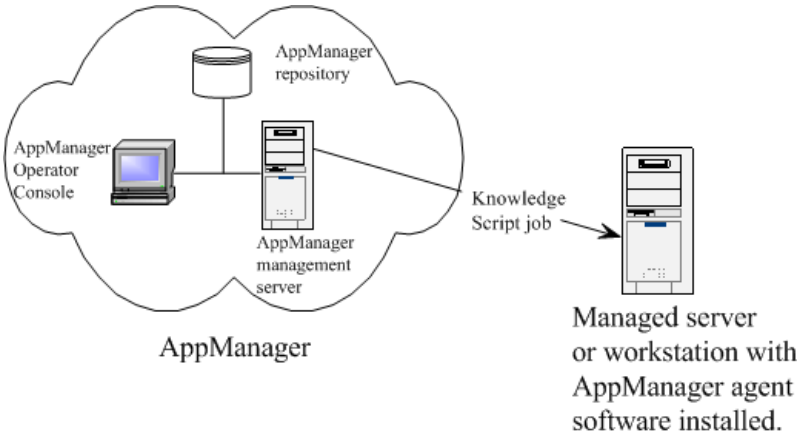
Overview of AppManager

If you are planning to use the NetIQOLE object to automate AppManager tasks, you should already be familiar with Knowledge Scripts, their script parameters, and jobs. Here is a review of the AppManager architecture and the process of job creation using a graphical user interface—either the Operator Console or the Web Console.

The NetIQOLE object is used to access the AppManager repository. In the following discussion, pay special attention to the role of the repository in creating and running jobs.

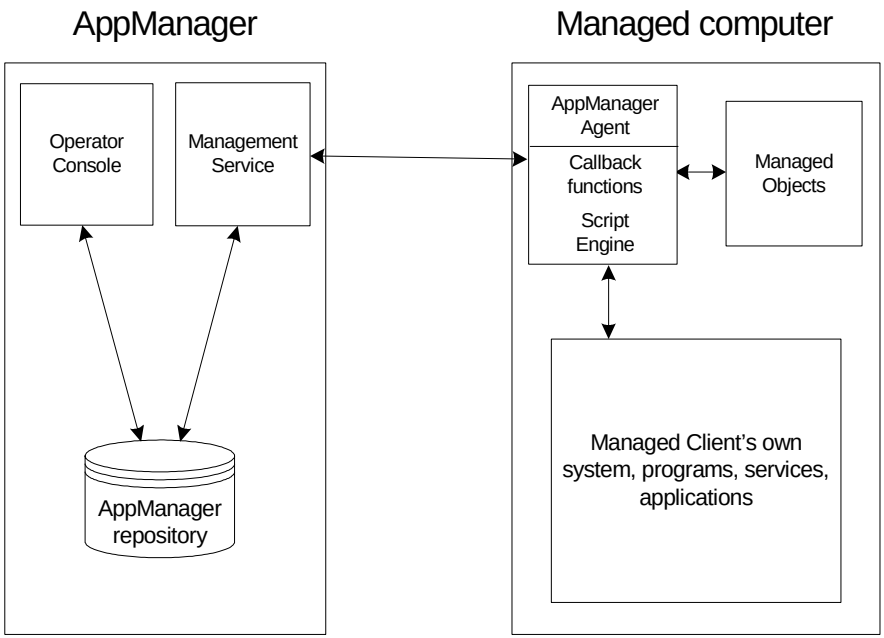
AppManager architecture

The AppManager Operator Console (or Web Console) creates Knowledge Script jobs to be run by the AppManager agent on the target computer. A simplified view is this:



The following drawing shows a more detailed view of the AppManager architecture to explain how Knowledge Scripts are processed and run. The drawing does not represent the only possible AppManager configuration—for example, the three components

shown as **AppManager** can be on the same server, as shown, but they do not need to be. Also, the components on the **Managed computer** have been simplified somewhat to facilitate discussion.



AppManager components

1. The Operator Console (or Web console) is the user interface for AppManager, and connects to the AppManager repository.

   Note The AppManager repository is called "QDB" by default, although it can be given any name during installation.

2. The repository is very important—it is the center of the AppManager world. Virtually everything that is involved in a running job is stored there. The repository server provides a central

store of information that includes Knowledge Scripts, events, graphs, and jobs (instances of running Knowledge Scripts). The job tables include the various pieces of running scripts, and other information such as scheduling.

3 The Management Service is responsible for polling the repository for newly created jobs and copying the job information to the AppManager agents on managed systems. The Management Service is also responsible for forwarding the events and data generated by jobs from the agents back into the AppManager repository.

Managed computer components

1 The AppManager agent performs a variety of tasks:

- It runs scripts (jobs).
- It has a local repository where it stores scripts, schedules, and actions.
- It communicates with the AppManager management server.

2 The managed objects are installed on the managed computer along with the AppManager agent, and are called by the scripts being run by the agent. They are COM objects or Perl modules containing methods that are specific to particular applications and are used to retrieve information about the monitored system or application that the script cannot obtain for itself.

Running Knowledge Scripts

These are the steps that the script undergoes when it is converted from a Knowledge Script stored in the AppManager repository to a job running on a target computer:

1 In the Operator Console, a user chooses a Knowledge Script, drags it, and drops it on the target object.

2 The **Properties** dialog box opens.

3. The user sets script parameters, execution schedule, actions, and so forth—or accepts the defaults. Then the user clicks **OK** to close the dialog box.

4. The Operator Console fills in values for the object types.

5. The Operator Console creates an instance of the script (a job) in the repository. The job is an instance of the script that includes the user defined script parameter values, the schedule, the object types, and so forth. This final script has all script parameters and object types defined as *constants* in BasicScript and *variables* in VBScript and Perl.

6. The Management Service copies the job to the AppManager agent on the target computer to be run there. Scheduling information (not part of the script) is also copied to the agent, as is information about Actions to perform.

All of the information about the job is also *retained in the AppManager repository*, along with pointers to any action scripts that are to be run on the AppManager management server.

Automating AppManager

As you can see from the prevous discussion:

- Knowledge Scripts are stored in the AppManager repository.
- All of the parameters, schedules, object types, and so forth of running jobs are stored in the repository.
- Event information returned by AppManager agents is stored in the repository.
- Data collected and returned by AppManager agents is stored in the repository.

Therefore, if you can access the AppManager repository programmatically, you can automate:

- The selection of Knowledge Scripts.
- The creation of jobs.

- The setting of job parameters, schedules, and so forth.
- The handling of events.
- The creation of graphs.
- Etc.

In other words, you can write programs that run AppManager as an alternative to using the Operator Console or Web Console.

AppManager command-line scripting is supported through an OLE automation object called NetIQOLE. The NetIQOLE automation object uses the ODBC SQL server driver to connect to the SQL Server where an AppManager repository has been installed.

NetIQOLE

NetIQOLE is intended for system administrators and advanced users who want to run AppManager from a command-line script. This guide assumes that you are familar with the AppManager Operator Console, Visual Basic programming, and common programming practices.

For more information about the Operator Console, refer to the *User Guide*.

Interfaces

NetIQOLE includes these interfaces:

- OLE automation-compatible interfaces for use with languages such as VBScript, Java, Active Server Pages (ASP), Visual Basic for Applications (VBA), and C++.
- Easy-to-use set of OLE interfaces that provide access to most AppManager data and functionality.

System requirements

NetIQOLE has the following system requirements:

- Windows NT 4.0 or later.
- NetIQ AppManager repository Version 4.0 or later.
- Should be registered on the computer where you use it.

Restrictions

NetIQOLE is only compatible with 32-bit developer tools including:

- Visual Basic 4.0 or later
- Microsoft Visual C++
- Visual Basic for Applications
- VBScript, JScript, etc.

Security

AppManager security is implemented through SQL Server. NetIQOLE user accounts require the same permissions to complete tasks as AppManager Operator Console user accounts.

If the user account does not have permission for a particular operation, the action will fail.

Command-line script interpreter

`netiqcmd.exe` is a command-line utility that provides a simple scripting host environment for NetIQOLE. This utility uses Microsoft's ActiveX Scripting Engine for its language parser support. `netiqcmd.exe` can execute scripts with the VBS and JVS extensions.

Running scripts with netiqcmd.exe

The `netiqcmd.exe` utility determines which scripting engine to use based on the script extension. For example, running the script `myscript.vbs` initializes and runs the script using the VBScript engine. Running the script `myscript.jvs` initializes and runs the script using the JScript engine.

Run `netiqcmd.exe` from the directory that has the script you want to run using the following format: `netiqcmd <script name>`. For example:

```
netiqcmd ADDMACHINE.VBS
```

When you run `netiqcmd.exe` without the correct arguments for the script you have named, you will be prompted with a list of arguments in the command prompt window. For `ADDMACHINE.VBS`, these arguments are:

Arugment	How to set it
/USER	Type the user name of the SQL Server login account used to access the AppManager repository. For example: `netiqcmd ADDMACHINE.VBS /USER=SA` The default SQL Server user is the administrative user, `sa`.
/PASSWORD	Type the password for the SQL Server login account. For example: `netiqcmd ADDMACHINE.VBS /USER=SA /PASSWORD=NETIQ` If using the `sa` user, there is no default password.
/SERVER	Type the Windows NT or Windows 2000 name of the server where the AppManager repository you want to work with is installed. For example: `netiqcmd ADDMACHINE.VBS /USER=SA /SERVER=SHASTA` The default server name is the local computer name.
/DATABASE	Enter the name of the AppManager database, or repository, you want to work with. For example: `netiqcmd ADDMACHINE.VBS /USER=SA /DATABASE=QDB` The default AppManager database name is QDB.
/DEBUGGING	Set to true to display descriptive SQL Server errors if they occur. For example: `netiqcmd ADDMACHINE.VBS /USER=SA /DEBUGGING=TRUE` The default is false.

Arugment	How to set it
/NEWMACHINE	Enter the name of the computer you want to add to the repository. For examaple: `netiqcmd ADDMACHINE.VBS /USER=SA /NEWMACHINE=SHASTA`
/MACHLIST	Enter the path to a text file which contains a list of computers that you want to add to the repository. For example: `netiqcmd ADDMACHINE.VBS /USER=SA /MACHLIST=C:\MYTEMP\MACHLIST.TXT` This argument is useful when adding many computers at once.

Saving netiqcmd.exe output

You can save all output generated by `netiqcmd.exe` to a file by redirection using `> <FILENAME>.TXT`. For example:

```
netiqcmd CLOSEJOB.VBS /USER=SA /SERVER=SHASTA > OUTPUT.TXT
```

The output file is saved to the directory where the script is being run.

The WSCRIPT Object

`netiqcmd.exe` has its own small API that is available to all scripts which are launched from `netiqcmd.exe`. The `WSCRIPT` object is included with the API. When you use `netiqcmd.exe`, you do not need to create the `WSCRIPT` object because the object is created for you. The `WSCRIPT` object enables a script to complete several calls including returning name of a script, launching an application, and sending text to a text file.

The following `WSCRIPT` objects are available:

Object	What it does

Arguments	Returns an instance of the arguments object.
	`Set ArgsObject = WScript.Arguments`
	`for i = 0 to ArgsObject.Count - 1`
	`  ' ### Retrieve the parameter`
	`  strVal = ArgsObject.Item(i)`
	`next`
	
	Notes:
	• Count returns the number of arguments in the collection.
	• Length is the same as count method. Provided to be call compatible with WSH.
	• Item(index) returns the argument at a particular index.

ScriptFullName	Returns the full path name of the running script.
	`Wscript.Echo Wscript.ScriptFullName`
	Example return
	`G:\netiq\scripts\AckEvent.vbs`

FullName	Returns the full path name of the script host.
	`Wscript.Echo Wscript.FullName`
	Example return:
	`g:\NetIQ\bin\netiqcmd.exe`

Version	Returns the version of `netiqcmd.exe`.
	`Wscript.Echo “netiqcmd version: ” &`
	`Wscript.Version`

ScriptName	Returns the 8.3 name of the currently running script. `Wscript.Echo Wscript.ScriptName`
Name	Returns the script host friendly name "NETIQ Script Host." `Wscript.Echo Wscript.Name`

Object	What it does

ShellExecute	Launches an application
	`Const SW_HIDE = 0`
	`Const SW_SHOWNORMAL = 1`
	`Const SW_NORMAL = 1`
	`Const SW_SHOWMINIMIZED = 2`
	`Const SW_SHOWMAXIMIZED = 3`
	`Const SW_MAXIMIZE = 3`
	`Const SW_SHOWNOACTIVATE = 4`
	`Const SW_SHOW = 5`
	`Const SW_MINIMIZE = 6`
	`Const SW_SHOWMINNOACTIVE = 7`
	`Const SW_SHOWNA = 8`
	`Const SW_RESTORE = 9`
	`Const SW_SHOWDEFAULT = 10`
	`Const SW_MAX = 10`
	
	`operation = “open”`
	`file = “calc.exe”`
	`parameters = “”`
	`directory = “c:\”`
	`showCMD = SW_SHOWNORMAL`
	`WScript.ShellExecute operation, file, parameters, directory, showCMD`

Echo	Sends text to the default output device. `Wscript.Echo “Send some text to standard out”`
MsgBox	Displays a standard message box. `Wscript.MsgBox “Message”, “Title”`

InputBox(Title, Prompt, style)	Displays a message box that contains an edit control. Whatever is typed in the control is returned in this function. If the style is set to 1, the edit control becomes a password field.
	`pw = Wscript.InputBox(“AppManager”, “Please Enter Password:”, 1)`
	`Wscript.Echo pw`

Object	What it does

UTCToDATE	Converts a UTC value into a Visual Basic-style date value. All dates in NetIQOLE are returned in universally coordinated time (SQL server date/time). `netiqcmd.exe` provides a handy function for converting those values into OLE date/times that are easier to work with in a scripting environment.
	UTC stands for Universal Coordinated Time, which refers to the time as set by the World Time Standard. Previously referred to as Greenwich Mean time or GMT. Most date/time values which are returned by NetIQOLE are returned in this format. It will be necessary to use the UTCtoDATE function to convert them to a format which scripting can understand (OLE date/times).
	`dtemp = WScript.UTCtoDATE(7234355)`
	`dt = FormatDateTime(dtemp, vbShortDate)`
	`tm = FormatDateTime(dtemp, vbShortTime)`
	`WScript.Echo dt + “ “ + tm`

ReportEvent(eventStr)	Writes an event to the application event log.
	`Const EVENT_ERROR = 0`
	`Const EVENT_WARNING = 1`
	`Const EVENT_INFORMATION = 2`
	`Const EVENT_SUCCESS = 3`
	`Const EVENT_FAILURE = 4`
	
	`appName = “CreateJob Script”`
	`msgText = “This is a test”`
	`eventID = 1`
	`eventType = EVENT_INFORMATION`
	`Wscript.ReportEvent appName, msgTxt, eventID, eventType`

ReturnCode	Allows a script to get/set the return code that `netiqcmd.exe` exits with. This may be useful if you are using `netiqcmd.exe` from within a batch file and need to check the return code by using the batch files's ERRORLEVEL. `Wscript.ReturnCode = 5`

Developer’s kit sample scripts

The NetIQOLE Developer’s kit includes the following sample command scripts. You can use some of these scripts as is, but most of them require additional input.

Script	What it does
ACKEVENT.VBS	Changes the status of an existing event to acknowledged.
ADDMACHINE.VBS	Adds a new computer to the Master view.
ADDSERVERGROUP.VBS	Adds a new group to the Master view.
CLOSEEVENT.VBS	Changes the status of an existing event to closed.
CLOSEJOB.VBS	Changes the status of an existing job to closed.
CREATEJOB.VBS	Creates a new job from an existing Knowledge Script.
CREATENTCPURESOURCE.VBS	Demonstrates how to use NetIQOLE to create a new job and modify the elements such as scheduling, parameters, actions, and object matching.
DELETEEVENT.VBS	Deletes an existing event.
DELETEJOB.VBS	Deletes an existing job.
DUMPGRAPH.VBS	Dumps a specified graph's data to the default output device in a comma-delimited format.
LISTEVENTS.VBS	Demonstrates how to enumerate events. Also a good example for filters. This script can also list events by computer and by severity.
LISTJOBS.VBS	Demonstrates how to enumerate jobs.
LISTKSPROPS.VBS	Lists all the defaults of a Knowledge Script: schedule, parameters, actions, and object matching information.
STARTJOB.VBS	Changes the status of an existing job to running.
STOPJOB.VBS	Changes the status of an existing job to stopped.
UPDATEEVENTCOMMENT.VBS	Demonstrates how to update the comment for a specified event.

Logon Defaults and Profiles

NetIQOLE provides two techniques for logging on to a SQL Server database: the Logon method and the ProfileLogon method.

Logon method

The Logon method is a simple method that uses four parameters:

Parameter	How to set it
MACHINE	Enter the name of the SQL Server where the AppManager repository is running.
DATABASE	Enter the name of the AppManager database, or repository, you want to work with. For example: `netiqcmd ADDMACHINE.VBS /USER=SA /DATABASE=QDB` The default AppManager database name is QDB.
USER	Type the user name of the SQL Server login account used to access the AppManager repository. For example: `netiqcmd ADDMACHINE.VBS /USER=SA` The default SQL Server user is the administrative user, `sa`.
PASSWORD	Type the password for the SQL Server login account. For example: `netiqcmd ADDMACHINE.VBS /USER=SA /PASSWORD=NETIQ` If using the `sa` user, there is no default password.

If any of these parameters are missing, NetIQOLE provides appropriate defaults. For example, if the server name is not specified, NetIQOLE looks for the computer key listed in the Default Logon section of a `NETIQ.INI` file in the `C:\WINNT` directory (you may need to create this file). If no computer key is available, NetIQOLE uses the local computer name.

You can also provide these parameters on the command line of the script.

Here is an example of the Default Logon section of a `NETIQ.INI` file:

```
[Default Logon]
MACHINE=SHASTA
```

```
DATABASE=QDB
USER=SA
PASSWORD=NETIQ
```

You can also use the NTAuthentication parameter to enable logons from trusted connections. To use this feature, set the NTAuthentication parameter to **true** prior to invoking the Logon method.

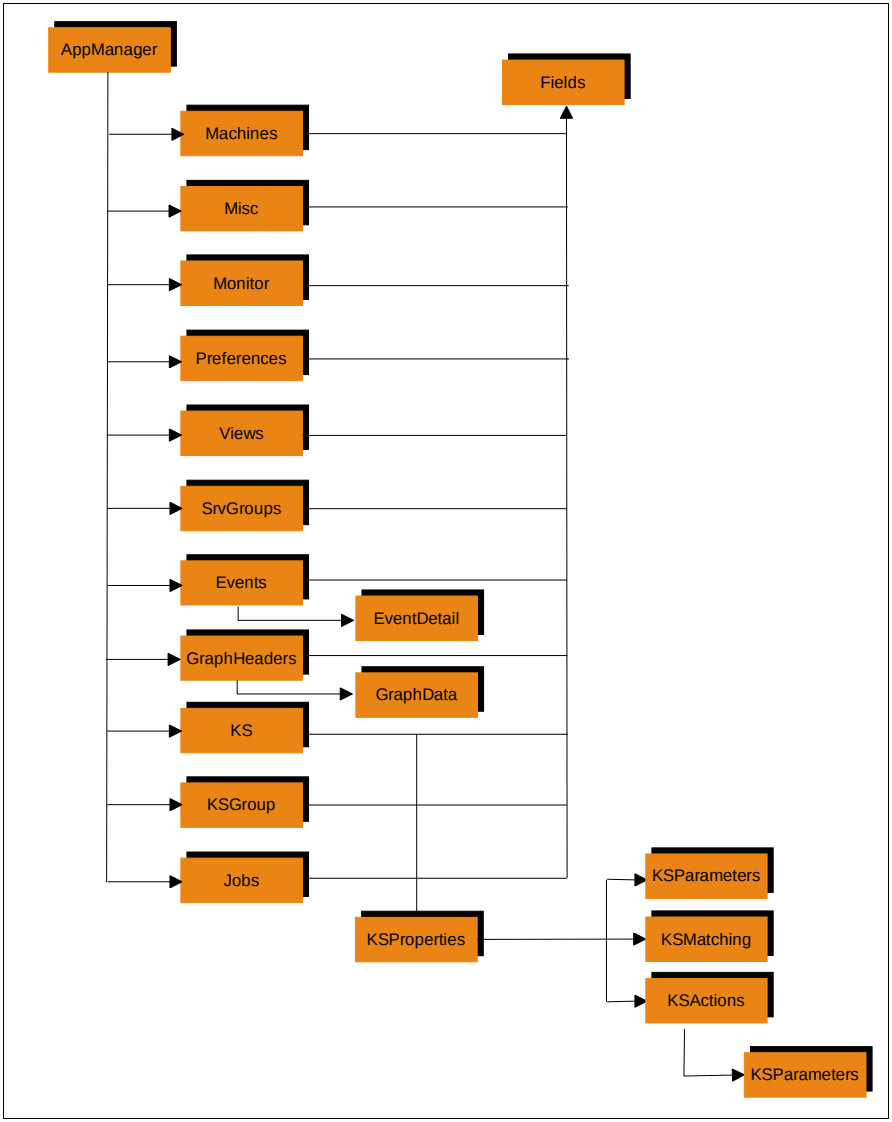
ProfileLogon method

The ProfileLogon method only uses the User parameter. The profile name is the name of the section in a `NETIQ.INI` file that contains the required logon parameters. This allows you to set up multiple logon profiles and easily switch between them. These profiles take the same form as the Default Logon profile, but the name of the section can be anything.

```
[Profile1]
DATABASE=QDB
USER=SA
PASSWORD=NETIQ
```

A full profile (including passwords) is not intended for environments where security is a concern. Profiles are provided only as a convenience. If security is important, consider using only a partial profile (for example, one that specifies just the MACHINE and DATABASE parameters).

NetIQOLE Object Model



Using filters to select a subset of data

Some NetIQOLE objects enable you to view data by directly querying the AppManager repository. Each object has a default query associated with it.

These data objects have additional built-in queries, called **filters**, which you can use to select a subset of data. Some filters require parameters to run properly.

Setting a new filter changes the record count and resets the current position to the first record in selected data.

The netiqole_constants.h header file lists the types of recordset filters and their required parameters. The macro names indicate to which object these filters can be applied to.

The following objects include filters:

- Machines
- Monitor
- Views
- SrvGroups
- Events
- EventDetails
- GraphHeaders
- GraphData
- KS
- KSGroup
- Jobs

Note The ability to retrieve data through NetIQOLE works for most purposes. However, keep in mind that reading data in NetIQOLE is not intended to be a high-performance operation.

Creating jobs

This topic is for users who are experienced Visual Basic programmers and understand the basics of running AppManager jobs. Consult *Developing Custom Knowledge Scripts* for insight into the way Knowledge Scripts are structured.

Creating a job with default parameters

There are several ways you can use NetIQOLE to create jobs. The simplest is to run the `CreateJob.vbs` script. For example, from the command line, run:

```
netiqcmd CREATEJOB.VBS /KSNAME=NT_CPUResource
                       /TARGET=STARGATE
```

This creates a job on the computer named STARGATE using the `NT_CPUResource` Knowledge Script. The job uses all the defaults for the `NT_CPUResource` Knowledge Script. NetIQOLE automatically finds all matching objects on the computer named STARGATE and uses them in the job.

When creating jobs, you do not need to change the Knowledge Script's parameters if the default values fit your needs. Unless otherwise specified, NetIQOLE automatically uses a Knowledge Script's default values when starting new jobs.

The `CreateJob.vbs` script will suffice if you are not interested in changing the default schedules, parameters, etc. If you do not want to accept the defaults, you will need a more complex script.

Creating a job with your own parameters

NetIQOLE provides the functionality for creating new jobs from existing Knowledge Scripts while modifying the scripts properties. NetIQOLE supports these Knowledge Script job creation and modification features:

- Creating a job from any existing Knowledge Script

- Modifying schedules
- Modifying parameters
- Selecting actions
- Modifying action parameters
- Modifying object matching information
- Modifying advanced options
- Parsing Knowledge Scripts

A step-by-step example

This example describes the programming steps required to create a job and alter its properties before running it. The sample code in this example is derived from the `CREATENTCPURESOURCE.VBS` script.

Creating a new job:

1. First, create an instance of the NetIQOLE object to use for tasks such as logging on. Creating the object returns a reference to the AppManager object.

   ```
   ' ### CREATE NetIQOLE OBJECT
   set AppObject = CreateObject(“NetIQOLE.APPMANAGER”)
   ```

   The AppManager object is the core object in NetIQOLE; it handles logging on to the AppManager repository.

2. Call the Logon method and verify that no errors were returned.

   ```
   ' ### ATTEMPT TO LOGON ON TO THE REPOSITORY
   WScript.Echo “Connecting to repository...”
   AppObject.Logon server, database, username, password
   If NOT AppObject.Error then
      ‘ ### At this point we should be connected to the
      repository
   End If
   ```

3 After you have successfully logged on, you can begin creating a new job. Creating a new job in NetIQOLE is similar to starting a new job in the AppManager Operator Console.

In the Operator Console, you pick a Knowledge Script from the Knowledge Script tab and drop it onto a computer in the TreeView pane. Similarly, with NetIQOLE you locate the Knowledge Script you want to run, and then make a call that tells NetIQOLE to run the Knowledge Script on a computer.

In NetIQOLE, you use the KS object; to start this process. The KS object; displays a list of all known (or checked-in) Knowledge Scripts. You can perform a search within the KS object; for a Knowledge Script and then set the internal NetIQOLE record pointer to the Knowledge Script you want to start.

The following example demonstrates how to obtain a list from the KS object; and use it to locate a Knowledge Script.

```
ksName = “NT_CPUResource”

' ### REQUEST THE KS OBJECT FROM THE CORE APPOBJECT

Set KSObject = AppObject.KS
KSObject.MoveFirst

' ### SEEK TO THE CORRECT KS
colIndex = KSObject.Fields.IndexFromName(“Name”)
KSObject.Search colIndex, ksName
If NOT KSObject.Error then
   ' ### At this point you have located our KS
End If
```

4 After locating the Knowledge Script, call the CommitKS method to confirm the use of this Knowledge Script. Then invoke the AddMachName method with the name of the computer where you want to run the Knowledge Script.

To run a Knowledge Script on multiple computers, call AddMachName method multiple times passing in the computer names.

Finally, use the ParseMultiMachs method to retrieve the Knowledge Script from the AppManager repository. It parses the Knowledge Script and pulls out the parameters, scheduling information, default actions, and object matching information, and advanced options information. The Parse method makes this information available in several subobjects, such as KSProperties, KSParameters, KSActions, etc. If the ParseMultiMachs method is successful, it returns a KSProperties object. From theKSProperties object., all other subobjects can be derived.

```
targetComputer = “STARGATE”
KSObject.CommitKS()
If KSObject.AddMachName (targetComputer) Then
   Set KSProperties = KSObject.ParseMultiMachs ()
End If
```

5 AppManager Knowledge Scripts contain parameters that enable you to control how the Knowledge Script runs. If the default parameters fit your needs, it is not necessary to modify the Knowledge Script before starting the job. However, you might want to change these defaults to better monitor your environment.

The KSParameters object is equivalent to the Values tab in the Knowledge Script Properties dialog box of the AppManager Operator Console. Use the KSParameters object to view and set specific parameters. Here is an example of how to modify parameters:

```
' ### Set up some KS parameters
Set KSParameters = KSProperties.Parameters
retCode = SetParameter(KSParameters, “DO_EVENT”, “y”)
retCode = SetParameter(KSParameters, “DO_DATA”, “n”)
retCode = SetParameter(KSParameters, “TH_UTIL”, “28”)
retCode = SetParameter(KSParameters, “Severity”, “3”)
```

Note Do not request subobjects such as Parameters, Actions, etc. more than once. It is best to do this once and pass the reference to any functions.

The parameter names you see using the KSParameters object are

not the same as they appear in the Values tab in the Operator Console. The Operator Console displays a more user-friendly version of the parameter names. To find the code name of a Knowledge Script parameter, look at the KPP section of the Knowledge Script.

This example shows that the parameter code names are constants and the user-friendly name shown in the Values tab follows in a comment (using a special syntax):

```
'### Begin KPP Section
const DO_EVENT = “y” '[M<Event? (y/n)>M][T<string,1,'
',”yn”>T]
const DO_DATA = “n” '[M<Collect Data?(y/
n)>M][T<string,1,' ',”yn”>T]
const TH_UTIL = 90 ' [M<CPU user
utilization>M][T<integer,1,' ',0,100,%>T]
const TH_PROC = 80 ' [M<Number of
processes>M][T<integer,1,' ',0,9999,#>T]
const TH_THREAD = 500 ' [M<Number of
threads>M][T<integer,1,' ',0,9999,#>T]
const TH_INTR = 600 ' [M<Interrupts per
second>M][T<integer,1,' ',0,9999,int/sec>T]
const Severity = 5 '[M<Event severity>M][T<integer,1,'
',1,40,SevLevel>T]
const AKPID = “AKP_NULL” ' [M<Action taken>M]
'### End KPP Section
```

The `CREATENTCPURESOURCE.VBS` script includes a function called SetParameter. In your own scripts, you can reuse this function to eliminate the work of looking up parameters you want to set. This function searches the KSParameters object for the name of the parameter you specified and, if it finds the parameter, it changes the parameter's value. All of the searching is done in-memory; it is a very fast operation which does not require any round trips to the AppManager repository.

```
Function SetParameter(ByVal KSParameters, ByVal
ParmName, ByVal newValue)
   retCode = FALSE
   ' ### It's always safest to make sure there are
```

```
   parameters present
   ' ### before accessing the parameters array.
      parmCount = KSParameters.Count
      If parmCount > 0 Then
         nParameterIndex =
         KSParameters.SearchByName(ParmName)
         If nParameterIndex > - 1 Then
            KSParameters.Value(nParameterIndex) =
            CStr(newValue)
            retCode = TRUE
         End If
      End If
   SetParameter = retCode
End Function
```

6 Modifying a job's schedule is perhaps the most complicated process in NetIQOLE. There are many methods and properties involved, plus several scheduling modes (Run Once, Daily, Monthly, etc).

The simplest type of job scheduling is one that runs once. The CREATENTCPURESOURCE.VBS script contains examples of how to each scheduling mode. All of the scheduling methods exist within the KSProperties object..

The following example sets the job to run once at a specified date and time.

```
Sub SetRunOnceScheduling(ByVal KSProperties)
   bStartNow = FALSE
   startDate =
   KSProperties.TranslateToNetIQDate(1999,11,13)
   startTime =
   KSProperties.TranslateToNetIQTime(13,30,00)
   KSProperties.SetRunOnceScheduling bStartNow,
   startDate, startTime
End Sub
```

AppManager has its own proprietary date and time format. Use the TranslateToNetIQDate and TranslateToNetIQTime methods to convert standard dates and times to the AppManager format. The calls TranslateFromNetIQDate and TranslateFromNetIQTime convert AppManager dates and times to the standard OLE date and

time formats that are recognized by Visual Basic. Here is an example:

```
' ### The dates and times values returned from these
functions can be used
' ### with any standard Visual Basic/VBScript date/time
formatting function.
startDateStr =
KSProperties.TranslateFromNetIQDate(KSProperties.
StartDate)
WScript.Echo “Start Date: “ & startDateStr

startTimeStr =
KSProperties.TranslateFromNetIQTime(KSProperties.StartT
ime)
WScript.Echo “Start Time: “ & startTimeStr
```

The following is an example of setting up an interval schedule.

```
Sub SetRegularIntervalScheduling(ByVal KSProperties)
   bStartNow = FALSE
   startDate =
   KSProperties.TranslateToNetIQDate(1999,11,13)
   startTime =
   KSProperties.TranslateToNetIQTime(13,30,00)

   stopType = STOP_TIME_NEVER 'Set to STOP_TIME_ON if you
   want 'a specific end date & time
   endtDate =
   KSProperties.TranslateToNetIQDate(1999,11,15)
   endTime = KSProperties.TranslateToNetIQTime(13,45,00)

   everyXInterval = 5
   intervalUnit = INTERVAL_MINUTES

   Iteration = 0

   KSProperties.SetIntervalScheduling bStartNow,
   stopType,
   intervalUnit,
   everyXInterval, StartDate, StartTime, endDate,
   endTime, Iteration
End Sub
```

The most complex scheduling mode is monthly. Monthly scheduling has several substates, such as scheduling on certain days of the month (such as the 1st, 3rd, 5th, and so on). You can also set up monthly scheduling to run on the 1st and 3rd weekends or the 3rd and 4th Wednesdays of the month. Here are some examples of how to code these scheduling modes:

Setting a schedule for specific days of the month:

```
' ### Demonstrates how to set up specific days of the
month
Sub SetMonthlyScheduling1(ByVal KSProperties)
   bStartNow = FALSE
   'stopType = STOP_TIME_NEVER
   'endDate = STOP_NEVER_DATE

   stopType = STOP_TIME_ON
   startDate =
   KSProperties.TranslateToNetIQDate(1999,11,13)
   endDate =
   KSProperties.TranslateToNetIQDate(1999,11,15)

   everyXMonths = 2
   'dailyFreq = DAILY_FREQ_ONCE
   dailyFreq = DAILY_FREQ_EVERY

   ' ### Setup the daily frequency section
   everyXFreq = 8
   dailyFreqUnit = INTERVAL_SECONDS

   monthDaysBits = CLng(MONTHLY_DAY_13 + MONTHLY_DAY_15 +
   MONTHLY_DAY_19 + MONTHLY+DAY_31)
   monthlyUnit = MONTHLY_SUB_DAY

   startTime =
   KSProperties.TranslateToNetIQTime(11,30,00)
   endTime = KSProperties.TranslateToNetIQTime(11,45,00)

   KSProperties.SetMonthlyScheduling bStartNow,
   stopType,
   MonthlyUnit,
   dailyFreq, dailyFreqUnit, monthDaysBits, monthWeeks,
   everyXMonths,
   everyXFreq, startDate, endDate, startTime, EndTime
```

```
   KSProperties.StopType = stopType
End Sub
```

Setting a schedule for specific weekends of the month:

```
' ### Demonstrates how to set up specific weekends of the
month
Sub SetMonthlyScheduling2(ByVal KSProperties)
   bStartNow = FALSE
   ' ### How to set up Stop Time Never
   'stopType = STOP_TIME_NEVER
   'endDate = STOP_NEVER_DATE

   stopType = STOP_TIME_ON
   startDate =
   KSProperties.TranslateToNetIQDate(1999,11,13)
   endDate =
   KSProperties.TranslateToNetIQDate(1999,11,15)

   everyXMonths = 2

   'dailyFreq = DAILY_FREQ_ONCE
   dailyFreq = DAILY_FREQ_EVERY

   ' ### Setup the daily frequency section
   everyXFreq = 8
   dailyFreqUnit = INTERVAL_SECONDS

   monthWeeks = MONTHLY_WEEK_2ND + MONTHLY_WEEK_4TH
   monthlyUnit = MONTHLY_SUB_WEEKDAY

   startTime =
   KSProperties.TranslateToNetIQTime(11,30,00)
   endTime = KSProperties.TranslateToNetIQTime(11,45,00)

   KSProperties.SetMonthlyScheduling bStartNow,
   stopType,
   MonthlyUnit,
   dailyFreq, dailyFreqUnit, monthDaysBits, monthWeeks,
   everyXMonths,
   everyXFreq, startDate, endDate, startTime, EndTime
End Sub
```

7 When you call the ParseMultiMachs function in the KS object;, two items are specified: the Knowledge Script and the computer where it runs.

In AppManager, each Knowledge Script operates against specific objects on a computer (such as a CPU or a specific database). By default, a job runs against all objects that apply to the Knowledge Script. For example, a SQL Server Knowledge Script may run against databases 1, 2, and 3 because those databases are located on the same SQL Server. AppManager calls this process **matching**.

In the Operator Console, the Object tab in the Knowledge Script Properties dialog box shows you all the objects that match the Knowledge Script type. The Object tab enables you to control which items a Knowledge Script runs against.

You may want to configure a Knowledge Script to run against only selected databases rather than all databases. In NetIQOLE, use the KSMatching object to select objects for the job to run against. The KSMatching object enables you to list all objects that match the Knowledge Script type and then deselect specific objects. By default, all objects that can be matched are selected (set to TRUE).

The following example demonstrates how to deselect an item by setting it to FALSE.

```
Set KSMatching = KSProperties.Matching
retCode = SetMatching(KSMatching, “CPU”, FALSE)
```

The SetMatchingEx function is a reusable function that appears in the CREATENTCPURESOURCE.VBS script. It looks up the specified object's name and changes its selection state.

```
Function SetMatchingEx (ByVal KSMatching, ByVal
MachineName, ByValObjectName,ByVal bSelectState
   retCode = False
   machCount = KSMatching.GetMachineCount()

   For i = 0 to machCount - 1
      machNameStr = KSMatching.GetMachineName(i)
```

```
      If machNameStr = MachineName Then
         matchingCount = KSMatching.CountMultiMachs(i)
         If matchingCount > 0 Then
            For j = 0 to matchingCount - 1
               objNameStr =
               KSMatching.GetObjectNameMultiMachs (j, i)
               If ObjNameStr = ObjectName Then
                  KSMatching.SelectionStateMultiMachs (j, i)
                  = bSelectState
                  SetMatchingEx = True
                  Exit Function
               End If
            Next
         End If
      End If
   Next
   SetMatchingEx = retCode
End Function
```

8 AppManager uses an action Knowledge Script. An action Knowledge Script is a script that is run by another Knowledge Script that has raised an event. For example, if you set up a Knowledge Script to monitor CPU usage, you can configure that Knowledge Script to run an action Knowledge Script if CPU usage exceeds the threshold. You can set action Knowledge Scripts to send pages, e-mails, messages, and many other actions. Multiple actions are allowed.

In the following example, the Action_RunSql Knowledge Script is associated with the sample Knowledge Script. Action Knowledge Scripts have their own set of parameters that can be modified. If you want to modify the parameters of an Action Knowledge Script, get the KSParameters object from the KSActions object and pass in the action's index as a parameter.

```
' ### Set up an Action KS
Set KSActions = KSProperties.Actions
actIndex = AddActionKS (KSActions, "Action_RunSql",
ACTLOC_MS, "", ACTTYPE_NEWEVT, 0, ACTSCHED_ALWAYS)
If actIndex >= 0 Then
   ' ### We set the action KS - now get the parameters for
   this
   ' ### action KS and modify them.
```

```
   Set KSActionParameters = KSActions.Parameters(0)
   retCode = SetParameter(KSActionParameters, “SqlCmd”,
   “select * from Job”)
End If
```

The AddActionKS function is a reusable helper function provided in the CREATENTCPURESOURCE.VBS script. It sets which Action Knowledge Script is associated with your standard Knowledge Script. It also enables you to specify if the action runs on the managed computer where the job is running, on the computer where the AppManager management server is running, or on a proxy computer. The AddActionKS function requires the action type and action schedule to be defined.

```
Function AddActinsKS (ByVal KSActions, ByVal
strActionKS, ByVal actLocation, ByVal strProxy, ByVal
actType, ByVal nRepeatCnt, ByVal actSchedule)

   actIndex = KSActions.AddAction(strActionKS)
   If actIndex >= 0 Then
      KSActions.ActionLocation(actIndex) = actLocation
      if actLocation = LOC_PROXY Then
         KSActions.ActionHost(actIncex) = strProxy
      End If
      KSActions.ActionSchedule(actIndex) = actSchedule
   Else
      Wscript.Echo “ERROR: Unable to add action - “ &
      strActionKS
   End If
   SetActionKS = actIndex
End Function
```

9 The last configurable section of a Knowledge Script is the Advanced options. There are two separate categories of options: Event and Data options. Event options are applied when a Knowledge Script is set to generate events. Data options are applied when the Knowledge Script is set to collect data. The CREATENTCPURESOURCE.VBS script provides functions that contain examples of how to configure each of these options.

The following example shows how to enable event collapsing.

```
‘turn on event collapsing and set the time interval for
event
‘collapsing 20 minutes
SetEventCollapsingInterval KSProperties, 20

‘select to measure the time interval from the events most
‘recent occurance
SetUseLastEvent KSProperties, 1

Sub SetEventCollapsingInterval(ByVal KSProperties, ByVal
collapseTime)
‘Time Interval for Event Collapsing
‘if collapseTime = 0, then Event Collapsing will be
disabled
   KSProperties.CollapsingInterval = collapseTime
End Sub

Sub SetUseLastEvent (ByVal KSProperties, ByVal newValue)
‘Time interval measure from
‘Inital occurrance = 1, Most Recent occurrance = 0

   KSProperties.UseLastEvent = newValue
End Sub
```

10 You are almost ready to commit your changes and create a new job. Before committing your changes, however, you should run to checks to validate your code.

First, verify that a discovered object has been selected. If no objects have been selected, the Knowledge Script will fail. There is a reusable helper function in the `CREATENTCPURESOURCE.VBS` script that performs this check. The function enumerates the KSMatching object and returns a count of all objects that have been selected.

Second, verify that the schedule information is valid. There are several conditions in scheduling that could cause the Knowledge Script to fail. In the KSProperties object., NetIQOLE provides a method called ValidateScheduleTime. If there is an error, this method returns several codes that explain what is wrong.

```
Function ValidateScheduling(ByVal KSProperties)
   retCode = FALSE
```

```
   validateCode = KSProperties.ValidateScheduleTime

   Select Case validateCode
      Case SUCCESS_SCHEDULE
         retCode = TRUE
      Case ERROR_SCHEDULE_WEEKLY_NEED_DAY
      Case ERROR_SCHEDULE_MONTHLY_NEED_DAY
      Case ERROR_SCHEDULE_MONTHLY_NEED_WEEK
      Case ERROR_SCHEDULE_STARTDATE_ET_CURDATE
      Case ERROR_SCHEDULE_STARTTIME_ET_CURTIME
      Case ERROR_SCHEDULE_STARTDATE_LT_STOPDATE
      Case ERROR_SCHEDULE_STARTTIME_LE_STOPTIME
      Case ERROR_SCHEDULE_DAILY_STARTDATE_EQ_STOPDATE
      Case ERROR_SCHEDULE_DAILY_STARTTIME_EQ_STOPTIME
         retCode = FALSE
   End Select
   ValidateScheduling = retCode
End Function
```

After successfully passing these two checks, you can create a new job.

The method for creating a new job is in the KS object; it is called CreateNewJob. If you do not call the CreateNewJob method, all modifications to the job are lost. If called successfully, the CreateNewJob method returns a new AppManager job ID.

```
' ### Before committing changes - it is good to verify a
few things
selectedObjects = GetSelectedObjects(KSMatching)
If selectedObjects > 0 Then

   If ValidateScheduling(KSProperties) Then
   ' ### Commit the changes
   newJobID = KSObject.CreateNewJob()
      If KSObject.Error then
         WScript.Echo “ERROR: Cannot create new job”
      Else
         WScript.Echo “SUCCESS: Job #” & newJobID & “
         created
         on computer: “ & targetComputer
      End if
   Else
      WScript.Echo “ERROR: There is a problem with the
```

```
      scheduling”
    End If
Else
   WScript.Echo “ERROR: There are no selected objects to
   run
   this KS against”
End If
```

Error codes

The following table lists the possible error codes returned by the LastError property on any NetIQOLE object.

Constant	Value
NetIQOLE_SUCCESS	1
NetIQOLE_UNKNOWN_ERROR	2
NetIQOLE_INVALID_ARGUMENT	3
NetIQOLE_NO_MEMORY	4
NetIQOLE_FALSE	5
NetIQOLE_NOT_IMPLEMENTED	6
NetIQOLE_FAILED	7
NetIQOLE_RECORDSET_NOT_INIT	8
NetIQOLE_FAILED_LOGON	9
NetIQOLE_OUTOFBOUNDS	10
NetIQOLE_INVALID_FILTER	11
NetIQOLE_NOTENOUGHRIGHTS	12
NetIQOLE_LICENSE_EXPIRE	13
NetIQOLE_LICENSE_INVALID	14
NetIQOLE_LICENSE_WARNING	15
NetIQOLE_QDB_VERSION_OLD	16
NetIQOLE_QDB_VERSION_NEW	17
NetIQOLE_MC_VERS3_NOTSUPPORTED	18

Constant	Value
NetIQOLE_MC_VERS4_NOTSUPPORTED	19
NetIQOLE_STANDARD_EXCEPTION	100

Programming tips

Here are some tips to help you when working with NetIQOLE.

- Requesting job-related subobjects
- Scope rules
- Using the Logoff method

Requesting job-related subobjects

When creating new jobs and calling the KSParameters object , KSActions object , or KSMatching object , you should call them only one time each.

```
Function SetParameter(ByVal KSProperties, ByVal ParmName, ByVal
newValue)

Set KSParameters = KSProperties.Parameters  ‘ ### INCORRECT
retCode = FALSE

' ### It is always safest to make sure there are parameters
present
' ### before accessing the parameters array.
parmCount = KSParameters.Count
If parmCount > 0 Then
   nParameterIndex = KSParameters.SearchByName(ParmName)
   If nParameterIndex > - 1 Then
      KSParameters.Value(nParameterIndex) = CStr(newValue)
      retCode = TRUE
   End If
End If

SetParameter = retCode

End Function
```

Requesting an object multiple times will reinitialize the KSParameters object each time resulting in the loss of your changes.

Here is an example of the correct way to request the KSParameters object . Request the KSParameters object one time and then pass it to the SetParameter call.

```
' ### Set up some KS parameters
Set KSParameters = KSProperties.Parameters
retCode = SetParameter(KSParameters, “DO_EVENT”, “y”)
retCode = SetParameter(KSParameters, “DO_DATA”, “n”)
retCode = SetParameter(KSParameters, “TH_UTIL”, “28”)
retCode = SetParameter(KSParameters, &”Severity”, “3”)
```

Scope rules

Be careful to refer to NetIQOLE objects only when they are available within the current scope. Trying to use an object that has gone out of scope can cause a crash.

```
Sub Func1(ByVal AppObject)
   Set JobsObject = AppObject.Jobs
   NumObjects = JobsObject.Count   ‘### Works Fine here
End Func1

AppObject.Logon server, database, username, password
   If NOT AppObject.Error then
      Func1 AppObject
      NumJobs = JobsObject.Count  ‘### Object defined out of
      scope will crash
   End If
```

Using the Logoff method

After you use the Logoff method to drop the connection to the repository, any call to cached subobjects (jobs, events, etc.) will fail. Be careful not to call any of these functions again. If you need to re-request the objects, log on again first.

```
AppObject.Logon server, database, username, password
If NOT AppObject.Error then
   Set JobsObject = AppObject.Jobs
```

```
   AppObject.Logoff

   ‘ ### BAD Jobs is no longer valid – will crash!
   NumJobs = JobsObject.Count
End If
```

Chapter 2

Object reference table

The NETIQOLE automation object is a set of hierarchical objects. The AppManager object is the root object from which all other objects can be obtained.

The following table summarizes the objects provided by the NETIQOLE automation object.

Object	Description
AppManager object	Main point of entry for NETIQOLE. The AppManager object contains logon methods and provides access to all sub-objects.
EventDetail object	Represents the detail for a particular event. You can also use this object to update an event comment.
Events object	Represents the set of events in the AppManager repository.
Fields object	Represents the schema of a recordset. Use this object to enumerate fields, discover size and type, and obtain field IDs.
GraphData object	Represents the actual data points for a specified graph data header
GraphHeaders object	Represents the set of Graph data items in the AppManager repository.
Jobs object	Represents the jobs in the AppManager repository.
KS object	Represents the list of Knowledge Scripts checked into the AppManager repository.
KSActions object	Represents the action to be taken by a particular job. Use this object to specify which action Knowledge Script is run and to change the action Knowledge Script's parameters.

Object	Description
KSMatching object	Represents the list of items that a Knowledge Script operates against. Items can be selected or deselected. By default, all matching items are selected.
KSGroup object	
KSParameters object	Represents the list of parameters for a given Knowledge Script. Use this object to enumerate and modify the parameters.
KSProperties object	The object returned when a job or Knowledge Script is parsed. Use this object to get and set schedule information. This object also returns the KSParameters object and the KSActions object.
Machines object	Represents the list of known computers.
Misc object	Provides miscellaneous utility functions.
Monitor object	Represents the list of computers with open events that fall within a specified event severity range.
Preferences object	Contains methods and properties used to get and set AppManager repository preferences.
SrvGroups object	Represents the list of server groups.
Views object	Represents the list of all discovered views.

Chapter 3

AppManager object

The main point of entry for NETIQOLE. The AppManager object contains logon methods and provides access to all sub-objects.

Use this object to logon to the AppManager repository. Once logged on, requests to other objects will be granted.

This section covers the following topics:

- AppManager properties
- AppManager methods

AppManager properties

The following properties are available for the AppManager object.

Property	Description
AppName	Get or set the application name.
Component	Sets the component ID for the application logging on.
Debugging	If TRUE the script will display descriptive SQL errors in a messagebox; the default is FALSE.
Error	Indicates if the last operation resulted in an error.
Events	Returns a reference to the Events object.
GraphHeaders	Returns a reference to the GraphHeaders object.
Jobs	Returns a Jobs object reference.
KS	Returns a KS object reference.
KSGroups	Returns a KSGroups object reference.
LastError	Indicates the error code of the last operation.
LicenseExpiration	Indicates the UI expiration date in UTC time.
Machines	Returns a reference to the Machines object.

Property	Description
Monitor	Returns a reference to the Monitor object.
NTAuthentication	Sets flag to use Windows NT Authentication in mixed mode.
Preferences	Returns a reference to the Preferences object.
QDB	Returns the name of the database where the user is currently logged on.
Server	Returns the name of the computer where the user is currently logged on.
SQLModeName	Returns the name of current mode of the SQL server.
SQLModeValue	Returns a number indicating the current mode of the SQL server.
Username	Returns the name of the user who is currently logged on.
Version	Returns the version number of the NETIQOLE object.
Views	Returns a reference to the Views object.

AppName

Get or set the application name.

Example

```
Property AppName () As String
```

Parameters

Return values

A string value representing the application name.

Remarks

Component

Sets the component ID for the application logging on.

Example

```
Property Component() As Long
```

Parameters

Return values

The component ID for the application logging on.

Remarks

Debugging

If TRUE, the script displays descriptive SQL errors in a messagebox; the default is FALSE.

Example

```
Property Debugging() As BOOL
```

Sample code

```
AppObject.Debugging = TRUE
```

Parameters

Return values

A boolean value to switch error messages on and off.

Error

Indicates if the last operation resulted in an error.

Example

```
Property Error() As BOOL
```

Parameters

Return values

A boolean value to indicate an error.

Remarks

Read-only attribute.

Events

Returns a reference to the Events object.

Example

```
Property Events() As Object
```

Parameters

Return values

An Events object reference.

Remarks

Read-only attribute.

GraphHeaders

Returns a reference to the GraphHeaders object.

Example

```
Property GraphHeaders() As Object
```

Sample code

```
set GraphsObject = AppObject.GraphHeaders
```

Parameters

Return values

A GraphHeaders object reference.

Remarks

Read-only attribute.

Jobs

Returns a Jobs object reference.

Example

```
Property Jobs() As Object
```

Parameters

Return values

A Jobs object reference.

Remarks

Read-only attribute.

KS

Returns a KS object reference.

Example

```
Property KS() As Object
```

Parameters

Return values

A KS object reference.

Remarks

Read-only attribute.

KSGroups

Returns a KSGroups object reference.

Example

```
Property KSGroups() As Object
```

Parameters

Return values

A KSGroups object reference.

Remarks

Read-only attribute.

LastError

Indicates the error code of the last operation.

Example

```
Property LastError() As Integer
```

Parameters

Return values

A number indicating the error code of the last operation.

Remarks

Read-only attribute.

LicenseExpiration

Indicates the UI expiration date in UTC time.

Example

```
Property LicenseExpiration() As Long
```

Parameters

Return values

UTC value of the UI expiration date.

Remarks

Read-only attribute.

Machines

Returns a reference to the Machines object.

Example

```
Property Machines() As Object
```

Parameters

Return values

A Machines object reference.

Remarks

Read-only attribute.

Monitor

Returns a reference to the Monitor object.

Example

```
Property Monitor() As Object
```

Parameters

Return values

A Monitor object reference.

Remarks

Read-only attribute.

NTAuthentication

Sets flag to use Windows NT Authentication in mixed mode.

Example

```
Property NTAuthentication() As BOOL
```

Sample code

```
AppObject.NTAuthentication = TRUE
```

Parameters

Return values

A boolean value to indicate using Windows NT Authentication in mixed mode.

Remarks

Read-only attribute.

Preferences

Returns a reference to the Preferences object.

Example

```
Property Preferences() As Object
```

Parameters

Return values

A Preferences object reference.

Remarks

Read-only attribute.

QDB

Returns the name of the database to which the user is currently logged on.

Example

```
Property QDB() As String
```

Parameters

Return values

A string value of the database name.

Remarks

Read-only attribute.

Server

Returns the name of the computer where the user is currently logged on.

Example

```
Property Server() As String
```

Parameters

Return values

A string value of the server name.

Remarks

Read-only attribute.

SQLModeName

Returns the name of current mode of the SQL server.

Example

```
Property SQLModeName() As String
```

Parameters

Return values

String value of the name of current mode of the SQL server.

Remarks

Read-only attribute.

SQLModeValue

Returns a number indicating the current mode of the SQL server.

Example

```
Property SQLModeValue() As Integer
```

Parameters

Return values

A number indicating the current mode of the SQL server.

Remarks

Read-only attribute.

Username

Returns the name of the user who is currently logged on.

Example

```
Property Username() As String
```

Parameters

Return values

A string value of the current user name.

Remarks

Read-only attribute.

Version

Returns the version number of the NETIQOLE object.

Example

```
Property Version() As String
```

Parameters

Return values

The current NETIQOLE version number.

Remarks

Read-only attribute.

Views

Returns a reference to the Views object.

Example

```
Property Views() As Object
```

Parameters

Return values

A Views object reference.

Remarks

Read-only attribute.

AppManager methods

The following methods are available for the AppManager object.

Methods	Descriptions
EncryptedLogon	Logs on using an AppManager supplied password.
Logoff	Logs off of the current repository.
Logon	Logs on to the specified AppManager repository.
ProfileLogon	Logs on to NETIQOLE using a profile.

EncryptedLogon

Logs on using an AppManager-supplied password.

Example

```
Sub EncryptedLogon(SERVERNAME As String,QDB As
String,Username As String,PASSWORD As String)
```

Parameters

- SERVERNAME - name of server
- QDB - name of the AppManager database
- Username - user name
- PASSWORD - password

Return values

Remarks

Logoff

Logs off from the current repository.

Example

```
Sub Logoff()
```

Sample code

```
AppObject.Logoff
```

Parameters

Return values

Remarks

Logon

Logs on to the specified AppManager repository.

Example

```
Sub Logon(SERVERNAME As String,QDB As String,Username As
String,PASSWORD As String)
```

Parameters

- SERVERNAME - name of server
- QDB - name of the AppManager database
- Username - user name
- PASSWORD - password

Sample code

```
server   = "stargate"
database = "QDB"
```

```
username = "uname"
password = "netiq"
AppObject.Logon server, database, username, password
If AppObject.Error Then MsgBox "Logon failed!"
End if
```

Return values

Remarks

ProfileLogon

Logs on to NETIQOLE using a profile.

Example

```
Sub ProfileLogon(PROFILE As String)
```

Parameters

- PROFILE - the profile header name

Return values

Remarks

Chapter 4

EventDetail object

This object represents the detail for a particular event. You can also use this object to update an event comment.

This section covers the following topics:

- EventDetail properties
- EventDetail methods

EventDetail properties

The following properties are available for the EventDetail object.

Property	Description
ActionCount	Retrieves the number of actions associated with current event
ActionEndTime	Retrieves the action's end time at a particular index
ActionMsg	Retrieves the action's message at a particular index
ActionName	Retrieves the action's name at a particular index
ActionStartTime	Retrieves the action's start time at a particular index
ActionStatus	Retrieves the action's status at a particular index
AutoRefresh	A flag indicating whether the recordset will be requeried after any operation which could affect the recordset (useful for batch operations).
BOF	Indicates that the current record position is before the first record.
Count	Returns the number of records in a recordset.
EOF	Indicates that the current record position is after the last record
Error	Indicates if the last operation resulted in an error.

Property	Description
EventLongMsg	Retrieves the event's message.
Fields	Returns an instance of a field's object.
LastError	Indicates the error code of the last operation.
SortIndex	Specifies which column in a recordset should be sorted on.
View	Sets or gets the view ID which will be used in view-specific objects.

ActionCount

Retrieves the number of actions associated with current event.

Example

```
Property ActionCount As Integer
```

Parameters

Return values

The number of actions associated with current event.

Remarks

Read-only attribute.

ActionEndTime

Retrieves the action's end time at a particular index.

Example

```
Property ActionEndTime(index as Integer) As Date
```

Parameters

- index - Index of the action

Return values

The action’s end time.

Remarks

Read-only attribute.

ActionMsg

Retrieves the action's message at a particular index.

Example

```
Property ActionMsg(index as Integer) As String
```

Parameters

- index - Index of the action

Return values

The action’s message.

Remarks

Read-only attribute.

ActionName

Retrieves the action’s name at a particular index.

Example

```
Property ActionName(index as Integer) As String
```

Sample code

```
Set EvtDetailObject = EventObject.Detail
numActions = EvtDetailObject.ActionCount()
For ix = 0 to numActions-1
   Wscript.Echo "Action = " &
   EvtDetailObject.ActionName(ix)
Next
```

Parameters

- index- Index of the action

Return values

The action’s name.

Remarks

Read-only attribute.

ActionStartTime

Retrieves the action's start time at a particular index.

Example

```
Property ActionStartTime(index as Integer) As Date
```

Parameters

- index - Index of the action

Return values

The action’s start time.

Remarks

Read-only attribute.

ActionStatus

Retrieves the action's status at a particular index.

Example

```
Property ActionStatus(index as Integer) As Long
```

Sample code

```
Set EvtDetailObject = EventObject.Detail
numActions = EvtDetailObject.ActionCount()
```

```
For ix = 0 to numActions-1
   Wscript.Echo "Status = " &
   EvtDetailObject.ActionStatus(ix)
Next
```

Parameters

- index- Index of the action

Return values

The action's status.

Remarks

Read-only attribute.

AutoRefresh

A flag indicating whether the recordset will be requeried after any operation which could affect the recordset (useful for batch operations).

Example

```
Property AutoRefresh() As BOOL
```

Parameters

Return values

A boolean expression that determines whether to automatically requery the recordset.

Remarks

BOF

Indicates that the current record position is before the first record.

Example

```
Property BOF() As BOOL
```

Parameters

Return values

A boolean expression.

Remarks

Read-only attribute.

Count

Returns the number of records in a recordset.

Example

```
Property Count() As Long
```

Parameters

Return values

The number of records in a record set.

Remarks

Read-only attribute.

EOF

Indicates that the current record position is after the last record.

Example

```
Property EOF() As BOOL
```

Parameters

Return values

A boolean expression.

Remarks

Read-only attribute.

Error

Indicates if the last operation resulted in an error.

Example

```
Property Error() As BOOL
```

Parameters

Return values

A boolean expression to indicate if an error occurred.

Remarks

Read-only attribute.

EventLongMsg

Retrieves the event's message.

Example

```
Property EventLongMsg As String
```

Sample code

```
Set EvtDetailObject = EventObject.Detail
Wscript.Echo "Long Event Msg = &
   " &EvtDetailObject.EventLongMsg
```

Parameters

Return values

The event’s long message.

Remarks

Fields

Returns an instance of a field's object.

Example

```
Property Fields() As Object
```

Parameters

Return values

An instance of a field's object.

Remarks

Read-only attribute.

LastError

Indicates the error code of the last operation.

Example

```
Property LastError() As Integer
```

Parameters

Return values

The error code of the last operation.

Remarks

Read-only attribute.

SortIndex

Specifies which column in a recordset should be sorted on.

Example

```
Property SortIndex() As Integer
```

Parameters

Return values

The current sort index number.

Remarks

View

Sets or gets the view ID which will be used in view-specific objects.

Example

```
Property View() As Integer
```

Parameters

Return values

The number ID of the view to limit by.

Remarks

EventDetail methods

The following methods are available for the EventDetail object.

Method	Description
GetField	Retrieves the data for a particular column in a record.
JumpAbsolute	Repositions the record pointer to the specified record number
JumpRelative	Repositions the record pointer x number of spaces from where it was last
MoveFirst	Moves the record position to the first record in the recordset.
MoveLast	Moves the record position to the last record in the recordset.

Method	Description
MoveNext	Moves the record position to the record after the current record.
MovePrevious	Moves the record position to the record before the current record.
Refresh	Refreshes the current recordset.
Search	A non case-sensitive search on a particular column.
UpdateComment	Updates the comment field for a particular event

GetField

Retrieves the data for a particular column in a record.

Example

```
Sub GetField(column As Integer) As String
```

Parameters

- column - The column of the desired field

Return values

The data for a particular column in a record.

Remarks

JumpAbsolute

Repositions the record pointer to the specified record number.

Example

```
Sub JumpAbsolute(offset As Long)
```

Parameters

Return values

Remarks

JumpRelative

Repositions the record pointer x number of spaces from where it was last.

Example

```
Sub JumpAbsolute(offset As Long)
```

Parameters

Return values

Remarks

MoveFirst

Moves the record position to the first record in the recordset.

Example

```
Sub MoveFirst()
```

Parameters

Return values

Remarks

MoveLast

Moves the record position to the last record in the recordset.

Example

```
Sub MoveLast()
```

Parameters

Return values

Remarks

MoveNext

Moves the record position to the record after the current record.

Example

```
Sub MoveNext()
```

Parameters

Return values

Remarks

MovePrevious

Moves the record position to the record before the current record.

Example

```
Sub MovePrevious()
```

Parameters

Return values

Remarks

Refresh

Refreshes the current recordset.

Example

```
Sub Refresh()
```

Parameters

Return values

Remarks

Search

A non case-sensitive search on a particular column.

Example

```
Sub Search(column As Integer,searchStr As String)
```

Parameters

- column - Number indicating which column to search on
- searchStr - The string to search for

Return values

Remarks

UpdateComment

Updates the comment field for a particular event.

Example

```
Sub UpdateComment(strComment As String)
```

Parameters

- strComment - The new comment

Return values

Remarks

Chapter 5

Events object

This object represents the set of events in the AppManager repository.

This section covers the following topics:

- Events properties
- Events methods

Events properties

The following properties are available for the Events object.

Property	Description
AutoRefresh	A flag indicating whether the recordset will be requeried after any operation which could affect the recordset (useful for batch operations).
BOF	Indicates the current record position is before the first record
Count	Returns the number of records in a recordset
Detail	Retrieves an event detail object
EOF	Indicates the current record position is after the last record
Error	Indicates if the last operation resulted in an error
Fields	Returns an instance of a field's object
HasComment	Indicates whether an event specified by the current record position has a comment
LastError	Indicates the error code of the last operation
SortIndex	Specifies which column in a recordset should be sorted on
View	Sets/Gets the view ID which will be used in view-specific objects

AutoRefresh

A flag indicating whether the recordset will be requeried after any operation which could affect the recordset (useful for batch operations).

Example

```
Property AutoRefresh() As BOOL
```

Parameters

Return values

A boolean expression that determines whether to automatically requery the recordset.

Remarks

BOF

Indicates the current record position is before the first record.

Example

```
Property BOF() As BOOL
```

Parameters

Return values

A boolean expression.

Remarks

Read-only attribute.

Count

Returns the number of records in a recordset.

Example

```
Property Count() As Long
```

Parameters

Return values

Remarks

Detail

Retrieves an event detail object.

Example

```
Property Detail() As Object
```

Parameters

Return values

An EventDetail object reference with detailed information about the current event.

Remarks

EOF

Indicates the current record position is after the last record.

Example

```
Property EOF() As BOOL
```

Parameters

Return values

A boolean expression.

Remarks

Read-only attribute.

Error

Indicates if the last operation resulted in an error.

Example

```
Property Error() As BOOL
```

Parameters

Return values

A boolean expression to indicate if an error occurred.

Remarks

Read-only attribute.

Fields

Returns an instance of a field's object.

Example

```
Property Fields() As Object
```

Parameters

Return values

An instance of a field's object.

Remarks

Read-only attribute.

HasComment

Indicates whether an event specified by the current record position has a comment.

Example

```
Property HasComment() As BOOL
```

Parameters

Return values

A boolean value indicating wheter the event has a comment.

Remarks

Read-only attribute.

LastError

Indicates the error code of the last operation.

Example

```
Property LastError() As Integer
```

Parameters

Return values

The error number of the last operation.

Remarks

Read-only attribute.

SortIndex

Specifies which column in a recordset should be sorted on.

Example

```
Property SortIndex() As Integer
```

Parameters

Return values

The current sort index number.

Remarks

View

Sets/Gets the view ID which will be used in view-specific objects.

Example

```
Property View() As Integer
```

Parameters

Return values

The number ID of the view to limit by.

Remarks

Events methods

The following methods are available for the Events object.

Method	Description
AckEvent	Acknowledges an open event specified by the current record position
AckEventEx	Acknowledges an event specified by a particular event ID
AckMultiEvents	Acknowledges multiple open events with eventid's specified in the event buffer
AddToEventBuffer	Inserts an event ID into the event buffer for batch manipulation
CloseEvent	Closes an event specified by the current record position

Method	Description
CloseEventEx	Closes an event specified by a particular event ID
CloseMultiEvents	Closes multiple events with eventid's specified in the event buffer
CreateEvent	Creates an event record.
CreateEvent2	Creates an event record with object.
DeleteEvent	Deletes the event at the current record position - event must have a status of closed
DeleteEventEx	Deletes an event specified by a particular event ID - event must have a status of closed
DeleteMultiEvents	Deletes multiple events with eventid's specified in the event buffer - event must have a status of closed
GetField	Retrieves the data for a particular column in a record
JumpAbsolute	Repositions the record pointer to the specified record number
JumpRelative	Repositions the record pointer x number of spaces from where it was last
MoveFirst	Moves the record position to the first record in the recordset
MoveLast	Moves the record position to the last record in the recordset
MoveNext	Moves the record position to the record after the current record
MovePrevious	Moves the record position to the record before the current record
Refresh	Refreshes the current recordset
ResetEventBuffer	Removes all eventid's in the event buffer
Search	A non case-sensitive search on a particular column
SetFilter	Applies a filter to the current recordset

AckEvent

Acknowledges an open event specified by the current record position.

Example

```
Sub AckEvent()
```

Parameters

Return values

Remarks

AckEventEx

Acknowledges an event specified by a particular event ID.

Example

```
Sub AckEventEx(eventID As Long)
```

Sample code

```
eventID = 1
EventObject.AckEventEx(eventID)
if EventObject.Error then
   WScript.Echo "ERROR: Failed to acknowledge event"
else
   WScript.Echo "SUCCESS: Event Acknowledged!!!"
end if
```

Parameters

- eventID - The event ID

Return values

Remarks

AckMultiEvents

Acknowledges multiple open events with event IDs specified in the event buffer.

Example

```
Sub AckMultiEvents()
```

Parameters

Return values

Remarks

Event IDs are located in the event buffer.

AddToEventBuffer

Inserts an event ID into the event buffer for batch manipulation.

Example

```
Sub AddToEventBuffer(eventID As 4 byte unsigned int)
```

Parameters

- eventID - The event ID

Return values

Remarks

CloseEvent

Closes an event specified by the current record position.

Example

```
Sub CloseEvent()
```

Parameters

Return values

Remarks

CloseEventEx

Closes an event specified by a particular event ID.

Example

```
Sub CloseEventEx(eventID As Long)
```

Sample code

```
eventID = 1
EventObject.CloseEventEx(eventID)
If EventObject.Error then
   WScript.Echo "ERROR: Failed to Close Event"
Else
   WScript.Echo "SUCCESS: Event #" & eventID & " closed!!!"
End if
```

Parameters

- eventID - The event ID

Return values

Remarks

CloseMultiEvents

Closes multiple events with event IDs specified in the event buffer.

Example

```
Sub CloseMultiEvents()
```

Sample code

```
EventObject.AddToEventBuffer(3)
EventObject.AddToEventBuffer(4)
EventObject.AddToEventBuffer(5)

‘delete events in the event buffer
EventObject.CloseMultiEvents()
```

Parameters

Return values

Remarks

Event IDs are located in the event buffer.

CreateEvent

Creates an event record.

Example

```
Sub CreateEvent(machineName As String,eventMsg As
String,jobID As Long,occurTime As Long,severity As Long) As
Long
```

Parameters

- machineName - Name of the computer to which the event belongs
- eventMsg - Message of the event
- jobID - The job ID from which the event was generated
- occurTime - Time of occurrence in UTC format
- severity - Severity of the event

Return values

Remarks

CreateEvent2

Creates an event record with object.

Example

```
Sub CreateEvent2(machineName As String,eventMsg As
String,agentMsg As String,object As String,jobID As
Long,occurTime As Long,severity As Long) As Long
```

Parameters

- machineName - Name of the computer to which the event belongs
- eventMsg - Message of the event
- agentMsg - Message from the agent
- object - Name of the object on which the event occurred
- jobID - The job ID from which the event was generated

- occurTime - Time of occurrence in UTC format
- severity - Severity of the new event

Return values

Remarks

DeleteEvent

Deletes the event at the current record position. The event must have a status of closed.

Example

```
Sub DeleteEvent()
```

Parameters

Return values

Remarks

DeleteEventEx

Deletes an event specified by a particular event ID. The event must have a status of closed.

Example

```
Sub DeleteEventEx(eventID As Long)
```

Parameters

- eventID - The event ID

Return values

Remarks

DeleteMultiEvents

Deletes multiple events with event ID's specified in the event buffer. The events must have a status of closed.

Example

```
Sub DeleteMultiEvents()
```

Sample Code

```
EventObject.AddToEventBuffer(3)
EventObject.AddToEventBuffer(4)
EventObject.AddToEventBuffer(5)

‘delete events in the event buffer
EventObject.DeleteMultiEvents()
```

Parameters

Return values

Remarks

The event IDs are located in the event buffer.

GetField

Retrieves the data for a particular column in a record.

Example

```
Sub GetField(column As Integer) As String
```

Parameters

- column - The column of the desired field

Return values

The data for a particular column in a record.

Remarks

JumpAbsolute

Repositions the record pointer to the specified record number.

Example

```
Sub JumpAbsolute(offset As Long)
```

Parameters

Return values

Remarks

JumpRelative

Repositions the record pointer x number of spaces from where it was last.

Example

```
Sub JumpRelative(offset As Long)
```

Parameters

Return values

Remarks

MoveFirst

Moves the record position to the first record in the recordset.

Example

```
Sub MoveFirst()
```

Parameters

Return values

Remarks

MoveLast

Moves the record position to the last record in the recordset.

Example

```
Sub MoveLast()
```

Parameters

Return values

Remarks

MoveNext

Moves the record position to the record after the current record.

Example

```
Sub MoveNext()
```

Parameters

Return values

Remarks

MovePrevious

Moves the record position to the record before the current record.

Example

```
Sub MovePrevious()
```

Parameters

Return values

Remarks

Refresh

Refreshes the current recordset.

Example

```
Sub Refresh()
```

Parameters

Return values

Remarks

ResetEventBuffer

Removes all event IDs in the event buffer.

Example

```
Sub ResetEventBuffer()
```

Parameters

Return values

Remarks

Search

A non case-sensitive search on a particular column.

Example

```
Sub Search(column As Integer,searchStr As String)
```

Parameters

- column - Number indicating which column to search on
- searchStr - The string to search for

Return values

Remarks

SetFilter

Applies a filter to the current recordset.

Example

```
Sub SetFilter(filterType As Integer,parm1 As String,parm2 As
String)
```

Parameters

- filterType - ID of the filter

  1 = // |Filter by Parent Events

  2 = // |Filter by Child Events

  3 = // |No Filter

  4 = // machine|Filter by Machine

  5 = // severity|Filter by Severity

  6 = // machine|Filter by Child Events and Machine

  7 = // severity|Filter by Child Events and Severity

  8 = // machine,severity|Filter by Machine and Severity

  9 = // username|Filter by EventID

  10 = // start severity, end severity|Filter by Severity within range

- parm1 - Value of parameter 1 if the filter requires one or more parameters, otherwise pass in an empty string
- parm2 - Value of parameter 2 if the filter requires a second parameter, otherwise pass in an empty string

Return values

Remarks

Chapter 6

Fields object

This object represents the schema of a recordset. Use this object to enumerate fields, discover size and type, and obtain field IDs.

This section covers the following topics:

- Fields properties
- Fields methods

Fields properties

The following properties are available for the Fields object.

Property	Description
Count	Returns the number of fields in a recordset.
Error	Indicates if the last operation resulted in an error.
LastError	Indicates the error code of the last operation.

Count

Returns the fields in a recordset.

Example

```
Property Count() As Integer
```

Parameters

Return values

The number of fields in a recordset.

Remarks

Read-only attribute.

Error

Indicates if the last operation resulted in an error.

Example

```
Property Error() As BOOL
```

Parameters

Return values

A boolean value indicating if the last operation resulted in an error.

Remarks

Read-only attribute.

LastError

Indicates the error code of the last operation.

Example

```
Property LastError() As Integer
```

Parameters

Return values

The error number of the last operation.

Remarks

Read-only attribute.

Fields methods

The following methods are available for the Fields object.

Method	Description
GetIsNull	Determines if the specified field is NULL.
IndexFromName	Returns the field index based on a given field name.
Name	Returns the name of a particular field.
Size	Returns the maximum size of a particular field.
Sortable	Indicates whether a particular field is sortable.
Type	Returns the type of a particular field.

GetIsNull

Determines if the specified field is NULL.

Example

```
Sub GetIsNull(fldIndex As Integer) As BOOL
```

Parameters

- fldIndex - Index of a specific field

Return values

A boolean value that determines if the specified field is NULL.

Remarks

IndexFromName

Returns the field index based on a given field name.

Example

```
Sub IndexFromName(columnName As String) As Integer
```

Parameters

- columnName - The name of the field

Return values

The index to the field.

Remarks

Name

Returns the name of a particular field.

Example

```
Sub Name(fldIndex As Integer) As String
```

Parameters

- fldIndex - Index of a specific field

Return values

The name of the field.

Remarks

Size

Returns the maximum size of a particular field.

Example

```
Sub Size(fldIndex As Integer) As Integer
```

Parameters

- fldIndex - Index of a specific field

Return values

The maximum size of a particular field.

Remarks

Sortable

Indicates whether a particular field is sortable.

Example

```
Sub Sortable(fldIndex As Integer) As BOOL
```

Parameters

- fldIndex - Index of a specific field

Return values

A boonlean value indicating whether the field is sortable.

Remarks

Type

Returns the type of a particular field.

Example

```
Sub Type(fldIndex As Integer) As Integer
```

Parameters

- fldIndex - Index of a specific field

Return values

The type of a particular field.

Remarks

Chapter 7

GraphData object

This object represents the actual data points for a specified graph data header.

This section covers the following topics:

- GraphData properties
- GraphData methods

GraphData properties

The following properties are available for the GraphData object.

Property	Description
AutoRefresh	A flag indicating whether the recordset will be requeried after any operation which could affect the recordset (useful for batch operations).
BOF	Indicates that the current record position is before the first record.
Count	Returns the number of records in a recordset.
EOF	Indicates the current record position is after the last record.
Error	Indicates if the last operation resulted in an error.
Fields	Returns an instance of a field's object.
LastError	Indicates the error code of the last operation.
SortIndex	Specifies which column in a recordset should be sorted on.

AutoRefresh

A flag indicating whether the recordset will be requeried after any operation which could affect the recordset (useful for batch operations).

Example

```
Property AutoRefresh() As BOOL
```

Parameters

Return values

A boolean expression that determines whether to automatically requery the recordset.

Remarks

BOF

Indicates that the current record position is before the first record.

Example

```
Property BOF() As BOOL
```

Parameters

Return values

A boolean expression.

Remarks

Read-only attribute.

Count

Returns the number of records in a recordset.

Example

```
Property Count() As Long
```

Parameters

Return values

Remarks

EOF

Indicates the current record position is after the last record.

Example

```
Property EOF() As BOOL
```

Parameters

Return values

A boolean expression.

Remarks

Read-only attribute.

Error

Indicates if the last operation resulted in an error.

Example

```
Property Error() As BOOL
```

Parameters

Return values

A boolean expression to indicate if an error occurred.

Remarks

Read-only attribute.

Fields

Returns an instance of a field's object.

Example

```
Property Fields() As Object
```

Parameters

Return values

An instance of a field's object.

Remarks

Read-only attribute.

LastError

Indicates the error code of the last operation.

Example

```
Property LastError() As Integer
```

Parameters

Return values

The error number of the last operation.

Remarks

Read-only attribute.

SortIndex

Specifies which column in a recordset should be sorted on.

Example

```
Property SortIndex() As Integer
```

Parameters

Return values

The current sort index number.

Remarks

GraphData methods

The following methods are available for the GraphData object.

Method	Description
GetDetails	Returns the details of a data point.
GetField	Retrieves the data for a particular column in a record.
JumpAbsolute	Repositions the record pointer to the specified record number.
JumpRelative	Repositions the record pointer x number of spaces from where it was last.
MoveFirst	Moves the record position to the first record in the recordset.
MoveLast	Moves the record position to the last record in the recordset.
MoveNext	Moves the record position to the record after the current record.
MovePrevious	Moves the record position to the record before the current record.
Refresh	Refreshes the current recordset.
Search	A non case-sensitive search on a particular column.
SetFilter	Applies a filter to the current recordset.

GetDetails

Returns the details of a data point.

Example

```
Sub GetDetails() As String
```

Parameters

Return values

A string value of the details of the data point.

Remarks

GetField

Retrieves the data for a particular column in a record.

Example

```
Sub GetField(column As Integer) As String
```

Parameters

- column - The column of the desired field

Return values

The data for a particular column in a record.

Remarks

JumpAbsolute

Repositions the record pointer to the specified record number.

Example

```
Sub JumpAbsolute(offset As Long)
```

Parameters

Return values

Remarks

JumpRelative

Repositions the record pointer x number of spaces from where it was last.

Example

```
Sub JumpAbsolute(offset As Long)
```

Parameters

Return values

Remarks

MoveFirst

Moves the record position to the first record in the recordset.

Example

```
Sub MoveFirst()
```

Parameters

Return values

Remarks

MoveLast

Moves the record position to the last record in the recordset.

Example

```
Sub MoveLast()
```

Parameters

Return values

Remarks

MoveNext

Moves the record position to the record after the current record.

Example

```
Sub MoveNext()
```

Parameters

Return values

Remarks

MovePrevious

Moves the record position to the record before the current record.

Example

```
Sub MovePrevious()
```

Parameters

Return values

Remarks

Refresh

Refreshes the current recordset.

Example

```
Sub Refresh()
```

Parameters

Return values

Remarks

Search

A non-case-sensitive search on a particular column.

Example

```
Sub Search(column As Integer,searchStr As String)
```

Parameters

- column - Number indicating which column to search on
- searchStr - The string to search for

Return values

Remarks

SetFilter

Applies a filter to the current recordset.

Example

```
Sub SetFilter(filterType As Integer,parm1 As String,parm2 As
String)
```

Parameters

- filterType - ID of the filter

  0 = ‘ | Data points associated with current graph header

  1 = ‘ | Show all data points

2 = ‘ Data ID, Time| Shows all points for a particular Data ID & Time

3 = ‘ Data ID | Shows all points for a particular Data ID

- parm1 - Value of parameter 1 if the filter requires one or more parameters, otherwise pass in an empty string
- parm2 - Value of parameter 2 if the filter requires a second parameter, otherwise pass in an empty string

Return values

Remarks

Chapter 8

GraphHeaders object

The object represents the set of Graph data items in the AppManager repository.

This section covers the following topics:

- GraphHeaders properties
- GraphHeaders methods

GraphHeaders properties

The following properties are available for the GraphHeaders object.

Property	Description
AutoRefresh	A flag indicating whether the recordset will be requeried after any operation which could affect the recordset (useful for batch operations).
BOF	Indicates that the current record position is before the first record.
Count	Returns the number of records in a recordset.
Data	Returns the GraphData object pointer containing the current graphs data.
EOF	Indicates the current record position is after the last record.
Error	Indicates if the last operation resulted in an error.
Fields	Returns an instance of a field's object.
LastError	Indicates the error code of the last operation.
SortIndex	Specifies which column in a recordset should be sorted on.

AutoRefresh

A flag indicating whether the recordset will be requeried after any operation which could affect the recordset (useful for batch operations).

Example

```
Property AutoRefresh() As BOOL
```

Parameters

Return values

A boolean expression that determines whether to automatically requery the recordset.

Remarks

BOF

Indicates that the current record position is before the first record.

Example

```
Property BOF() As BOOL
```

Parameters

Return values

A boolean expression.

Remarks

Read-only attribute.

Count

Returns the number of records in a recordset.

Example

```
Property Count() As Long
```

Parameters

Return values

Remarks

Data

Returns the GraphData object pointer containing the current graph's data.

Example

```
Property Data() As Object
```

Sample code

```
datastreamID = 1

colIndex = GraphsObject.Fields.IndexFromName("DataID")
GraphsObject.Search colIndex, datastreamID

if NOT GraphsObject.Error then
   set GraphDataObject = GraphsObject.Data
End if
```

Parameters

Return values

The GraphData object reference containing the current graph's data.

Remarks

EOF

Indicates the current record position is after the last record.

Example

```
Property EOF() As BOOL
```

Parameters

Return values

A boolean expression.

Remarks

Read-only attribute.

Error

Indicates if the last operation resulted in an error.

Example

```
Property Error() As Bool
```

Parameters

Return values

A boolean expression to indicate if an error occurred.

Remarks

Read-only attribute.

Fields

Returns an instance of a field's object.

Example

```
Property Fields() As Object
```

Parameters

Return values

An instance of a field's object.

Remarks

Read-only attribute.

LastError

Indicates the error code of the last operation.

Example

```
Property LastError() As Integer
```

Parameters

Return values

The error number of the last operation.

Remarks

Read-only attribute.

SortIndex

Specifies which column in a recordset should be sorted on.

Example

```
Property SortIndex() As Integer
```

Parameters

Return alues

The current sort index number.

Remarks

GraphHeaders methods

The following methods are available for the GraphHeaders object.

Method	Description
DeleteGraph	Deletes graph data specified by the current record position.
DeleteGraphEx	Deletes graph data specified by a particular graph ID.
GetField	Retrieves the data for a particular column in a record.
JumpAbsolute	Repositions the record pointer to the specified record number.
JumpRelative	Repositions the record pointer x number of spaces from where it was last.
MoveFirst	Moves the record position to the first record in the recordset.
MoveLast	Moves the record position to the last record in the recordset.
MoveNext	Moves the record position to the record after the current record.
MovePrevious	Moves the record position to the record before the current record.
Refresh	Refreshes the current recordset.
Search	A non case-sensitive search on a particular column.
SetFilter	Applies a filter to the current recordset

DeleteGraph

Deletes graph data specified by the current record position.

Example

```
Sub DeleteGraph()
```

Parameters

Return values

Remarks

DeleteGraphEx

Deletes graph data specified by a particular graph ID.

Example

```
Sub DeleteGraphEx(graphID As Long)
```

Sample code

```
GraphID = 1
GraphObject.DeleteGraphEx graphID
```

Parameters

- graphID - The ID of the graph to delete

Return values

Remarks

GetField

Retrieves the data for a particular column in a record.

Example

```
Sub GetField(column As Integer) As String
```

Parameters

- column - The column of the desired field

Return values

The data for a particular column in a record.

Remarks

JumpAbsolute

Repositions the record pointer to the specified record number.

Example

```
Sub JumpAbsolute(offset As Long)
```

Parameters

Return values

Remarks

JumpRelative

Repositions the record pointer x number of spaces from where it was last.

Example

```
Sub JumpAbsolute(offset As Long)
```

Parameters

Return values

Remarks

MoveFirst

Moves the record position to the first record in the recordset.

Example

```
Sub MoveFirst()
```

Parameters

Return values

Remarks

MoveLast

Moves the record position to the last record in the recordset.

Example

```
Sub MoveLast()
```

Parameters

Return values

Remarks

MoveNext

Moves the record position to the record after the current record.

Example

```
Sub MoveNext()
```

Parameters

Return values

Remarks

MovePrevious

Moves the record position to the record before the current record.

Example

```
Sub MovePrevious()
```

Parameters

Return values

Remarks

Refresh

Refreshes the current recordset.

Example

```
Sub Refresh()
```

Parameters

Return values

Remarks

Search

A non-case-sensitive search on a particular column.

Example

```
Sub Search(column As Integer,searchStr As String)
```

Parameters

- column - Number indicating which column to search on
- searchStr - The string to search for

Return values

Remarks

SetFilter

Applies a filter to the current recordset.

Example

```
Sub SetFilter(filterType As Integer,parm1 As String,parm2 As
String)
```

Parameters

- filterType - ID of the filter

  0 = ‘Show all graphs

  1 = ‘Data ID | Shows all graph headers for a particular Data ID

  2 = ‘View ID | Shows all graph headers for a particular View ID

- parm1 - Value of parameter 1 if the filter requires one or more parameters, otherwise pass in an empty string
- parm2 - Value of parameter 2 if the filter requires a second parameter, otherwise pass in an empty string

Return values

Remarks

Chapter 9

Jobs object

This object represents the jobs in the AppManager repository.

This section covers the following topics:

- Jobs properties
- Jobs methods

Jobs properties

The following properties are available for the Jobs object.

Property	Description
AutoRefresh	A flag indicating whether the recordset will be requeried after any operation which could affect the recordset (useful for batch operations).
BOF	Indicates that the current record position is before the first record.
Count	Returns the number of records in a recordset.
EOF	Indicates the current record position is after the last record.
Error	Indicates if the last operation resulted in an error.
Fields	Returns an instance of a field's object.
LastError	Indicates the error code of the last operation.
SortIndex	Specifies which column in a recordset should be sorted on.
View	Sets/Gets the view ID which will be used in view-specific objects.
VersionCount	Get the number of versions available.
VersionID	Get/set the version ID currently selected.

AutoRefresh

A flag indicating whether the recordset will be requeried after any operation which could affect the recordset - useful for batch operations.

Example

```
Property AutoRefresh() As BOOL
```

Parameters

Return values

A boolean expression that determines whether to automatically requery the recordset.

Remarks

BOF

Indicates that the current record position is before the first record.

Example

```
Property BOF() As BOOL
```

Parameters

Return values

A boolean expression.

Remarks

A read-only attribute.

Count

Returns the number of records in a recordset.

Example

```
Property Count() As Long
```

Parameters

Return values

The number of records in a recordset.

Remarks

A read-only attribute.

EOF

Indicates the current record position is after the last record.

Example

```
Property EOF() As BOOL
```

Parameters

Return values

A boolean expression.

Remarks

A read-only attribute.

Error

Indicates if the last operation resulted in an error.

Example

```
Property Error() As BOOL
```

Parameters

Return values

A boolean expression to indicate if an error occurred.

Remarks

A read-only attribute.

Fields

Returns an instance of a field's object.

Example

```
Property Fields() As Object
```

Parameters

Return values

An instance of a field's object.

Remarks

A read-only attribute.

LastError

Indicates the error code of the last operation.

Example

```
Property LastError() As Integer
```

Parameters

Return values

The error number of the last operation.

Remarks

A read-only attribute.

SortIndex

Specifies which column in a recordset should be sorted on.

Example

```
Property SortIndex() As Integer
```

Parameters

Return values

The current sort index number.

Remarks

View

Sets/Gets the view ID which will be used in view-specific objects.

Example

```
Property View() As Integer
```

Parameters

Return values

The number ID of the view to limit by.

Remarks

VersionCount

Get the number of versions available.

Example

```
Property VersionCount() As Integer
```

Parameters

Return values

The number of versions available.

VersionID

Get/set the version ID currently selected.

Example

```
Property VersionID() As 4 byte unsigned int
```

Parameters

Return values

The version ID currently selected.

Jobs methods

The following methods are available for the Jobs object.

Method	Description
AddChildrenJobs	Add a child job to an existing parent job.
AddMachName	Add a machine name to a list for running a job.
AddToJobBuffer	Inserts a job ID into the job buffer for batch manipulation.
CloseJob	Changes the status of the job at the current record position to closed.
CloseJobEx	Changes the status of a job to closed - specified by a particular job ID.
CommitParentJob	Commits the parent job for children jobs.
DeleteJob	Deletes the job at the current record position.
DeleteJobEx	Deletes a job specified by a particular job ID.

Method	Description
DeleteMultiJobs	Deletes multiple jobs with jobid's specified in the job buffer.
GetField	Retrieves the data for a particular column in a record.
GetMatching	Retrieves the matching object of the machines to start job on.
JumpAbsolute	Repositions the record pointer to the specified record number.
JumpRelative	Repositions the record pointer x number of spaces from where it was last.
MoveFirst	Moves the record position to the first record in the recordset.
MoveLast	Moves the record position to the last record in the recordset.
MoveNext	Moves the record position to the record after the current record.
MovePrevious	Moves the record position to the record before the current record.
Parse	Parses a job and returns a reference to the KSProperties object.
ParseForInsertion	Parses a parent job.
Refresh	Refreshes the current recordset.
ResetJobBuffer	Removes all jobid's in the job buffer.
Search	A non case-sensitive search on a particular column.
SetFilter	Applies a filter to the current recordset.
SetInfo	Commits any changes made to a particular job properties
StartJob	Changes the status of the job at the current record position to running.
StartJobEx	Changes the status of a job to running - specified by a particular job ID.
StartMultiJobs	Changes status of the multiple jobs to running with jobid's specified in the job buffer.
StopJob	Changes the status of the job at the current record position to stopped.

Method	Description
StopJobEx	Changes the status of a job to stopped - specified by a particular job ID.
VersionIDByIndex	Get the version id of a specified index.

AddChildrenJobs

Add a child job to an existing parent job.

Example

```
Sub AddChildrenJobs()
```

Sample code

```
JobsObject.CommitParentJob()
If NOT JobsObject.Error then
   numMachines = 0
   If JobsObject.AddMachName("STARGATE") Then
   numMachines = numMachines + 1
   End If

   If numMachines > 0 Then
      JobsObject.ParseForInsertion()
      JobsObject.AddChildrenJobs()
   End if
End If
```

Parameters

Return values

Remarks

This requires the job to already be parsed.

AddMachName

Add a machine name to a list for running a job.

Example

```
Sub AddMachName(machName As String) as BOOL
```

Parameters

- machName - Name of the computer to run job on

Return values

Boolean value indicating success/failure locating the computer in the repository.

Remarks

AddToJobBuffer

Inserts a job ID into the job buffer for batch manipulation.

Example

```
Sub AddToJobBuffer(jobID As 4 byte unsigned int)
```

Parameters

- jobID - The job ID

Return values

Remarks

CloseJob

Changes the status of the job at the current record position to closed.

Example

```
Sub CloseJob()
```

Parameters

Return values

Remarks

CloseJobEx

Changes the status of a job to closed - specified by a particular job ID.

Example

```
Sub CloseJobEx(jobID As Long)
```

Sample code

```
JobID = 123
JobObject.CloseJobEx(jobID)
If JobObject.Error then
   WScript.Echo "ERROR: Failed to close job"
Else
   WScript.Echo "SUCCESS: Job #" & jobID & " Closed!"
End if
```

Parameters

- jobID - The job ID

Return values

Remarks

CommitParentJob

Commits the parent job for adding child jobs.

Example

```
Sub CommitParentJob()
```

Parameters

Return values

Remarks

DeleteJob

Deletes the job at the current record position.

Example

```
Sub DeleteJob()
```

Parameters

Return values

Remarks

DeleteJobEx

Deletes a job specified by a particular job ID.

Example

```
Sub DeleteJobEx(jobID As Long)
```

Parameters

- jobID - The job ID

Return values

Remarks

DeleteMultiJobs

Deletes multiple jobs with job ID's specified in the job buffer.

Example

```
Sub DeleteMultiJobs()
```

Sample code

```
JobObject.AddToJobBuffer(1)
JobObject.AddToJobBuffer(2)
JobObject.AddToJobBuffer(3)
JobObject.DeleteMultiJobs()
```

Parameters

Return values

Remarks

Job IDs are located in the job buffer.

GetField

Retrieves the data for a particular column in a record.

Example

```
Sub GetField(column As Integer) As String
```

Parameters

- column - The column of the desired field

Return values

The data for a particular column in a record.

Remarks

GetMatching

Retrieves the matching object of the machines on which to start a job.

Example

```
Sub GetMatching() As Object
```

Parameters

Return values

Remarks

JumpAbsolute

Repositions the record pointer to the specified record number.

Example

```
Sub JumpAbsolute(offset As Long)
```

Parameters

Return values

Remarks

JumpRelative

Repositions the record pointer x number of spaces from where it was last.

Example

```
Sub JumpAbsolute(offset As Long)
```

Parameters

Return values

Remarks

MoveFirst

Moves the record position to the first record in the recordset.

Example

```
Sub MoveFirst()
```

Parameters

Return values

Remarks

MoveLast

Moves the record position to the last record in the recordset.

Example

```
Sub MoveLast()
```

Parameters

Return values

Remarks

MoveNext

Moves the record position to the record after the current record.

Example

```
Sub MoveNext()
```

Parameters

Return values

Remarks

MovePrevious

Moves the record position to the record before the current record.

Example

```
Sub MovePrevious()
```

Parameters

Return values

Remarks

Parse

Parses a job and returns a reference to the KSProperties object.

Example

```
Sub Parse() As Object
```

Parameters

Return values

A KSProperties object reference to modify the job properties.

Remarks

ParseForInsertion

Parses a parent job.

Example

```
Sub ParseForInsertion() As Object
```

Parameters

Return values

Remarks

Refresh

Refreshes the current recordset.

Example

```
Sub Refresh()
```

Parameters

Return values

Remarks

ResetJobBuffer

Removes all job IDs in the job buffer.

Example

```
Sub ResetJobBuffer()
```

Parameters

Return values

Remarks

Search

A non-case-sensitive search on a particular column.

Example

```
Sub Search(column As Integer,searchStr As String)
```

Parameters

- column - Number indicating which column to search on
- searchStr - The string to search for

Return values

Remarks

SetFilter

Applies a filter to the current recordset.

Example

```
Sub SetFilter(filterType As Integer,parm1 As String,parm2 As
String)
```

Parameters

- filterType - ID of the filter

  1 = // |Filter by Open Jobs

  2 = // |Filter by Closed Jobs

  3 = // |No Filter

  4 = // |Filter by Open Parent Jobs

  5 = // |Filter by Closed Parent Jobs

  6 = // |Filter by Parent Jobs

  7 = // |Filter by Open Children

  8 = // |Filter by Closed Children

  9 = // |Filter by Children

  10 = // machine|Filter by Machine

  11 = // job id|Filter by Job ID

12 = // |Filter by Running Jobs

13 = // |Filter by Stopped Jobs

14 = // ksname|Filter by ksname

15 = // Multiple filter options, for Reporting use only

- parm1 - Value of parameter 1 if the filter requires one or more parameters, otherwise pass in an empty string
- parm2 - Value of parameter 2 if the filter requires a second parameter, otherwise pass in an empty string

Return values

Remarks

SetInfo

Commits any changes made to a particular job property.

Example

```
Sub SetInfo()
```

Parameters

Return values

Remarks

Requires that the job has been parsed.

StartJob

Changes the status of the job at the current record position to running.

Example

```
Sub StartJob()
```

Parameters

Return values

Remarks

StartJobEx

Changes the status of a job to running, based on the job ID.

Example

```
Sub StartJobEx(jobID As Long)
```

Parameters

- jobIv - The job ID

Return values

Remarks

StartMultiJobs

Changes status of multiple jobs to running, based on job IDs specified in the job buffer.

Example

```
Sub StartMultiJobs()
```

Sample code

```
JobObject.AddToJobBuffer(1)
JobObject.AddToJobBuffer(2)
JobObject.AddToJobBuffer(3)
JobObject.StartMultiJobs()
```

Parameters

Return values

Remarks

Job IDs are located in the job buffer.

StopJob

Changes the status of the job at the current record position to stopped.

Example

```
Sub StopJob()
```

Parameters

Return values

Remarks

StopJobEx

Changes the status of a job to stopped, based on the job ID.

Example

```
Sub StopJobEx(jobID As Long)
```

Parameters

- jobID - The job ID

Return values

Remarks

StopMultiJobs

Changes status of the multiple jobs to stopped, based on job IDs specified in the job buffer.

Example

```
Sub StopMultiJobs()
```

Parameters

Return values

Remarks

Job IDs are stored in the job buffer.

VersionIDByIndex

Sub Get the version ID of a specified index.

Example

```
Sub VersionIDByIndex(versIndex As Integer) As 4 byte unsigned
int
```

Parameters

- versIndex - Index of the version ID

Return values

The version ID.

Remarks

Job IDs are stored in the job buffer.

Chapter 10

KS object

This object represents the list of Knowledge Scripts checked into the AppManager repository.

This section covers the following topics:

- KS properties
- KS methods

KS properties

The following properties are available for the KS object.

Properties	Description
AutoRefresh	A flag indicating whether the recordset will be requeried after any operation which could affect the recordset (useful for batch operations).
BOF	Indicates that the current record position is before the first record.
Count	Returns the number of records in a recordset.
EOF	Indicates the current record position is after the last record.
Error	Indicates if the last operation resulted in an error.
Fields	Returns an instance of a field's object.
MemberCount	Returns the number of members associated with the current KS Group.
MemberDesc	Get the description of a member KS inside a KS Group specified by an index.
MemberJobID	Returns the job id that was created corresponding the member KS that is specified by an index.
MemberMatching	Returns the KSMatching object reference of a member KS inside a KS Group specified by an index.

Properties	Description
MemberName	Get the name of a member KS inside a KS Group specified by an index.
LastError	Specifies which column in a recordset should be sorted on.
MemberVersionCount	Get the number of versions available.
MemberVersionID	Get/Set the version ID currently selected.
SortIndex	Specifies which column in a recordset should be sorted on.
VersionMask	Returns the versions supported by the KS.

AutoRefresh

A flag indicating whether the recordset will be requeried after any operation which could affect the recordset (useful for batch operations).

Example

```
Property AutoRefresh() As BOOL
```

Parameters

Return values

A boolean expression that determines whether to automatically requery the recordset.

Remarks

BOF

Indicates that the current record position is before the first record.

Example

```
Property BOF() As BOOL
```

Parameters

Return values

A boolean expression.

Remarks

A read-only attribute.

Count

Returns the number of records in a recordset.

Example

```
Property Count() As Long
```

Parameters

Return values

The number of records in a recordset.

Remarks

A read-only attribute.

EOF

Indicates the current record position is after the last record.

Example

```
Property EOF() As BOOL
```

Parameters

Return values

A boolean expression.

Remarks

A read-only attribute.

Error

Indicates if the last operation resulted in an error.

Example

```
Property Error() As BOOL
```

Parameters

Return values

A boolean expression to indicate if an error occurred.

Remarks

A read-only attribute.

Fields

Returns an instance of a field's object.

Example

```
Property Fields() As Object
```

Parameters

Return values

An instance of a field's object.

Remarks

A read-only attribute.

MemberCount

Returns the number of members associated with the current KS Group.

Example

```
Property MemberCount As Integer
```

Parameters

Return values

The number of members associated with the KS Group.

Remarks

MemberDesc

Get the description of a member KS inside a KS Group, specified by an index.

Example

```
Property MemberDesc(nIndex As Integer) As String
```

Parameters

- nIndex - The index value of the member KS

Return values

The description value of a member KS.

Remarks

A read-only attribute.

MemberJobID

Returns the job ID corresponding to the member KS specified by an index.

Example

```
Property MemberJobID(nIndex As Integer) As Integer
```

Parameters

- nIndex - The index value of the member KS

Return values

The job ID corresponding to the specified member KS index. The job ID is only valid after calling the CreateNewJob method.

Remarks

A read-only attribute.

MemberMatching

Returns the KSMatching object reference of a member KS in a KS Group, specified by an index.

Example

```
Property MemberMatching(nIndex As Integer) As Object
```

Parameters

- nIndex - The index value of the member KS

Return values

The KSMatching object reference to a member KS.

Remarks

A read-only attribute.

MemberName

Get the name of a member KS in a KS Group, specified by an index.

Example

```
Property MemberName(nIndex As Integer) As String
```

Sample code

```
numMembers = KSObject.MemberCount
for i = 0 to numMembers-1
   wscript.echo "member = " & KSObject.MemberName(i)
next
```

Parameters

- nIndex - The index value of the member KS

Return values

The name of the member KS.

Remarks

LastError

Indicates the error code of the last operation.

Example

```
Property LastError() As Integer
```

Parameters

Return values

The error number of the last operation.

Remarks

A read-only attribute.

MemberVersionCount

Get the number of versions available.

Example

```
Property MemberVersionCount(nIndex AS Integer) As Integer
```

Parameters

- nIndex - The index value of the member KS

Return values

The number of available versions of this member KS.

Remarks

A read-only attribute.

MemberVersionID

Get/Set the version ID currently selected.

Example

```
Property MemberVersionID(nIndex As Integer) As 4 byte
unsigned int
```

Parameters

- nIndex - The index value of the member KS

Return values

Remarks

The current version ID selected.

SortIndex

Specifies which column in a recordset should be sorted on.

Example

```
Property SortIndex() As Integer
```

Parameters

Return values

The current sort index number.

Remarks

VersionMask

Returns the versions supported by the KS.

Example

```
Property VersionMask() As 4 byte unsigned int
```

Parameters

Return values

The versions supported be the KS.

Remarks

A read-only attribute.

KS methods

The following methods are available for the KS object.

Method	Description
AddMachName	Add a machine object to run job on.
CommitKS	Commit to using this KS for creating jobs.
CreateNewJob	Applies any changes to the KSProperties object .
DoKSCheckin	Checks a KS in to the repository.
DoKSCheckout	Checks a KS out from the repository.

Method	Description
GetField	Retrieves the data for a particular column in a record.
GetKSIconID	Get the icon id for a KS.
IsKSBag	Returns TRUE if the KS is a KS Group.
JumpAbsolute	Repositions the record pointer to the specified record number.
JumpRelative	Repositions the record pointer x number of spaces from where it was last.
MemberVersionIDBy Index	Get the version id of a specified index.
MoveFirst	Moves the record position to the first record in the recordset.
MoveLast	Moves the record position to the last record in the recordset.
MoveNext	Moves the record position to the record after the current record.
MovePrevious	Moves the record position to the record before the current record.
Parse	Parses the current KS and returns the KSProperties object reference.
ParseKSBag	Parses the current KS Group and returns the number of members.
ParseMultiMachs	Parses the current KS to run on multiple machines and returns the KSProperties object reference.
Refresh	Refreshes the current recordset.
Search	A non case-sensitive search on a particular column.
SetFilter	Applies a filter to the current recordset.

AddMachName

Add a machine object on which to run the job.

Example

```
Sub AddMachName(targetMachine As String) As BOOL
```

Sample code

```
KsObject.CommitKS()
```

```
If NOT (KsObject.AddMachName("star") Then
   Wscript.Echo "Unable to locate target computer: " &
   "starg"
End If
```

Parameters

- targetMachine - Name of the machine on which to run job

Return values

A Boolean value indicating whether the machine was successfully added.

Remarks

Use this method only after invoking the CommitKS method.

CommitKS

Commit to using this KS for creating jobs.

Example

```
Sub CommitKS()
```

Parameters

Return values

Remarks

CreateNewJob

Applies any changes to the KSProperties object.

Example

```
Sub CreateNewJob() As Long
```

Parameters

Return values

The Job ID of the new job.

Remarks

The KS or KS Group must be parsed prior to calling this function.

DoKSCheckin

Checks a KS in to the repository.

Example

```
Sub DoKSCheckin(ksFileName as String, srcPath as String) As
Boolean
```

Parameters

- ksFileName - The file name of the KS
- srcPath - The source path to the file location

Return values

True for success, False for failure.

Remarks

DoKSCheckout

Checks a KS out from the repository.

Example

```
Sub DoKSCheckout(ksFileName as String, srcPath as String,
version as Long) As Boolean
```

Parameters

- ksFileName - The file name of the KS

- srcPath - The source path to the file location
- version - The version of the KS

Return values

True for success, False for failure.

Remarks

GetField

Retrieves the data for a particular column in a record.

Example

```
Sub GetField(column As Integer) As String
```

Parameters

- column - The column of the desired field

Return values

The data for a particular column in a record.

Remarks

GetKSIconID

Get the icon ID for a KS.

Example

```
Sub GetKSIconID(ksid As Long) As Long
```

Parameters

- ksid - The ID of the KS

Return values

The icon ID.

Remarks

IsKSBag

Returns TRUE if the KS is a KS Group.

Example

```
Sub IsKSBag() As Long
```

Parameters

Return values

A Boolean value indicating whether the KS is a KS Group.

Remarks

JumpAbsolute

Repositions the record pointer to the specified record number.

Example

```
Sub JumpAbsolute(offset As Long)
```

Parameters

Return values

Remarks

JumpRelative

Repositions the record pointer x number of spaces from where it was last.

Example

```
Sub JumpAbsolute(offset As Long)
```

Parameters

Return values

Remarks

MemberVersionIDByIndex

Get the version ID of a specified index.

Example

```
Sub MemberVersionIDByIndex(nIndex As Integer,versIndex As
Integer) As 4 byte unsigned int
```

Parameters

- nIndex - The index value of the member KS
- versIndex - The index value of the version

Return values

The version ID currently selected.

Remarks

MoveFirst

Moves the record position to the first record in the recordset.

Example

```
Sub MoveFirst()
```

Parameters

Return values

Remarks

MoveLast

Moves the record position to the last record in the recordset.

Example

```
Sub MoveLast()
```

Parameters

Return values

Remarks

MoveNext

Moves the record position to the record after the current record.

Example

```
Sub MoveNext()
```

Parameters

Return values

Remarks

MovePrevious

Moves the record position to the record before the current record.

Example

```
Sub MovePrevious()
```

Parameters

Return values

Remarks

Parse

Parses the current KS and returns the KSProperties object reference.

Example

```
Sub Parse(targetMachine As String) As Object
```

Parameters

- targetMachine - Name of the machine on which to run the job

Return values

A KSProperties object reference to allow for configuration of the KS.

Remarks

This function limits one computer per job.

ParseKSBag

Parses the current KS Group and returns the number of members.

Example

```
Sub ParseKSBag() As Integer
```

Sample code

```
If KSObject.IsKSBag() then
   if KSObject.ParseKSBag() <= 0 Then
      MsgBox "Failed to parse KS Group"
   End if
End if
```

Parameters

Return values

The number of member KSs parsed.

Remarks

ParseMultiMachs

Parses the current KS to run on multiple machines and returns the KSProperties object reference.

Example

```
Sub ParseMultiMachines() As Object
```

Sample code

```
Set KSProperties = KSObject.ParseMultiMachs()
If KSProperties.Error Then
   MsgBox "Failed to parse KS"
End if
```

Parameters

Return values

A KSProperties object reference to allow for configuration of the KS.

Remarks

Refresh

Refreshes the current recordset.

Example

```
Sub Refresh()
```

Parameters

Return values

Remarks

Search

A non-case-sensitive search on a particular column.

Example

```
Sub Search(column As Integer,searchStr As String)
```

Parameters

- column - Number indicating on which column to search
- searchStr - The string to search for

Return values

Remarks

SetFilter

Applies a filter to the current recordset

Example

```
Sub SetFilter(filterType As Integer,parm1 As String,parm2 As
String)
```

Parameters

- filterType - ID of the filter

  1 = ‘ group name|Filter by Group Name

  2 = ‘ |No Filter

  3 = ‘ kp id|Filter by Knowledge Script ID

  4 = ‘ kp name|Filter by Knowledge Script Name

5 = ‘ |Filter by Action KS's

- parm1 - Value of parameter 1 if the filter requires one or more parameters, otherwise pass in an empty string
- parm2 - Value of parameter 2 if the filter requires a second parameter, otherwise pass in an empty string

Return values

Remarks

Chapter 11

KSActions object

This object represents the action to be taken by a particular job. Use this object to specify which action Knowledge Script is run and to change the action Knowledge Script's parameters.

This section covers the following topics:

- KSActions properties
- KSActions methods

KSActions properties

The following properties are available for the KSActions object.

Property	Description
ActionHost	Returns the action's proxy machine name.
ActionLocation	Returns the action's location.
ActionSchedule	Returns the action's schedule.
ActionType	Returns the action's type.
Count	Returns the number of actions associated with a knowledge script or job.
Description	Returns the description of an action at index 0.
DescriptionEx	Returns the description of an action specified by an index.
Error	Indicates if the last operation resulted in an error.
IsMCLocalAction	Determines if an action is configured to run on the local managed computer.
LastError	Indicates the error code of the last operation.
Message	Returns the action message specified by an index.
Name	Returns the name of the action specified by an index.

Property	Description
Parameters	Returns a reference to the Parameters object.
RepeatCnt	Returns the repeat count.
Value	Returns the value of an action specified by the index; index should be 0.

ActionHost

Returns the action's proxy machine name.

Example

```
Property ActionHost(nIndex As Integer) As String
```

Parameters

- nIndex - Index of the parameter

Return values

The proxy computer name on which to run the action.

Remarks

ActionLocation

Returns the action's location.

Example

```
Property ActionLocation(nIndex As Integer) As Integer
```

Sample code

```
Set KSActions    = KSProperties.Actions()

actindex = KSActions.AddAction("Action_DosCommand")
If actindex < 0 Then
   Wscript.Echo "Error: Unable to add action"
End if
KSActions.ActionLocation(index1) = ACTLOC_MC
```

Parameters

- nIndex - Index of the parameter

Return values

- Available action locations:

  ACTLOC_MS = 1 'always at MS(DEFAULT)

  ACTLOC_MC = 2 'always at MC

  ACTLOC_PROXY = 3 'remote MC

Remarks

ActionSchedule

Returns the action's schedule.

Example

```
Property ActionSchedule(nIndex As Integer) As Integer
```

Parameters

- nIndex - Index of the parameter

Return values

- Available action locations:

  ACTSCHED_ALWAYS = 1 '(DEFAULT)

  ACTSCHED_WORKHOURS = 2

  ACTSCHED_NONWORKHOURS = 3

  ACTSCHED_WEEKDAYS = 4

  ACTSCHED_WEEKENDS = 5

  ACTSCHED_CUSTOM1 = 6

  ACTSCHED_CUSTOM2 = 7

ACTSCHED_CUSTOM3 = 8

ACTSCHED_CUSTOM4 = 9

ACTSCHED_CUSTOM5 = 10

ACTSCHED_CUSTOM6 = 11

ACTSCHED_CUSTOM7 = 12

ACTSCHED_CUSTOM8 = 13

ACTSCHED_CUSTOM9 = 14

ACTSCHED_CUSTOM10 = 15

Remarks

ActionType

Returns the action's type.

Example

```
Property ActionType(nIndex As Integer) As Integer
```

Sample code

```
Set KSActions    = KSProperties.Actions()

actindex = KSActions.AddAction("Action_DosCommand")
If actindex < 0 Then
   Wscript.Echo "Error: Unable to add action"
End if
KSActions.ActionType(index1) = ACTTYPE_REPEATEVT
```

Parameters

- nIndex - Index of the parameter

Return values

- Available action types:

  ACTTYPE_NEWEVT = 1 'upon a new event(DEFAULT)

ACTTYPE_REPEATEVT = 2 'upon a repeat event

Remarks

Count

Returns the number of actions associated with a knowledge script or job.

Example

```
Property Count() As Integer
```

Parameters

Return values

The number of actions.

Remarks

Read-only attribute.

Description

Returns the description of an action at index 0.

Example

```
Property Description As String
```

Parameters

Return values

A string value description of the first action.

Remarks

Read-only attribute.

DescriptionEx

Returns the description of an action specified by an index.

Example

```
Property DescriptionEx(nIndex As Integer) As String
```

Parameters

- nIndex - Index of the parameter

Return values

A string value description of the action.

Remarks

Read-only attribute.

Error

Indicates if the last operation resulted in an error.

Example

```
Property Error() As BOOL
```

Parameters

Return values

A boolean value indicating an error in the last operation.

Remarks

Read-only attribute.

IsMCLocalAction

Determines if an action is configured to run on the local managed computer.

Example

```
Property IsMCLocalAction(nIndex As Integer) As Long
```

Parameters

- nIndex - Index of the parameter

Return values

A boolean value indicating to run action on the local managed computer.

Remarks

LastError

Indicates the error code of the last operation.

Example

```
Property LastError() As Integer
```

Parameters

Return values

The error number of the last operation.

Remarks

Read-only attribute.

Message

Returns the action message specified by an index.

Example

```
Property Message(nIndex As Integer) As String
```

Parameters

- nIndex - Index of the parameter

Return values

The string value of the action message.

Remarks

Read-only attribute.

Name

Returns the name of the action specified by an index.

Example

```
Property Name(nIndex As Integer) As String
```

Parameters

Return values

String value of the action's name.

Remarks

Read-only attribute.

Parameters

Returns a reference to the Parameters object.

Example

```
Property Parameters(nIndex As Integer) As Object
actindex = KSActions.AddAction("Action_DosCommand")
If actindex < 0 Then
   Wscript.Echo ''Error: Unable to add action''
End If
KSActions.ActionType(index1) = ACTTYPE_REPEATEVT
KSActions.RepeatCnt(index1) = 10
```

Parameters

- nIndex - Index of the parameter

Return values

A KSParameters object reference for configuration of the action's parameters.

Remarks

RepeatCnt

Returns the repeat count.

Example

```
Property RepeatCnt(nIndex As Integer) As Long
```

Sample code

```
Set KSActions = KSProperties.Actions()
```

Parameters

- nIndex - Index of the parameter

Return values

The number of times an event is raised before running the action.

Remarks

Value

Returns the value of an action specified by the index; index should be 0.

Example

```
Property value(nIndex As Integer) As String
```

Parameters

- nIndex - Index of the parameter

Return values

The action value.

Remarks

KSActions methods

The following methods are available for the KSActions object.

Method	Description
AddAction	Returns the index to the new action
SetNewJobInterval	Resets the current job interval

AddAction

Returns the index to the new action

Example

```
Sub AddAction(lpszActionName As String) As Integer
```

Sample code

```
Set KSActions = KSProperties.Actions()
actindex = KSActions.AddAction("Action_DosCommand")
If actindex < 0 Then
   Wscript.Echo ''Error: Unable to add action''
End if
```

Parameters

- lpszActionName - The name of the action

Return values

The index of the action. -1 if failed to add action.

Remarks

SetNewJobInterval

Resets the current job interval.

Example

```
Sub SetNewJobInterval(nIndex As Integer,hour As Integer,min
As Integer,sec As Integer)
```

Parameters

- nIndex - Index of the action
- hour - The hour
- minutes - The minutes
- seconds - The seconds

Return values

Remarks

Chapter 12

KSGroup object

This section covers the following topics:

- KSGroup properties
- KSGroup methods

KSGroup properties

The following properties are available for the KSGroup object.

Property	Description
Description	Returns the description of the KS Group.
Type	Returns the type ID of the KS Group.

Description

Returns the description of the KS Group.

Example

```
Property Description As String
```

Parameters

Return values

A string value of the description.

Remarks

Type

Returns the type ID of the KS Group.

Example

```
Property Type As Integer
```

Parameters

- Possible type IDs

  KSTYPE_REGULAR = 0

  KSTYPE_DISCOVERY = 1

  KSTYPE_INSTALL = 2

Return values

Remarks

KSGroup methods

The following methods are available for the KSGroup object.

Method	Description
MemberCount	Returns the number of members in this KS Group.
MemberName	Returns the name of member specified by index.
ParseKSGroup	Parses the KS Group for its members.
ParseNewKSGroup	Parse to create a new KS Group.
RemoveMemberKS	Removes a member KS from the current KS Group.
SearchMemberIndex	Returns the index to a member.
UpdateAll	Commit all changes and save KS Group.

MemberCount

Returns the number of members in this KS Group.

Example

```
Sub MemberCount As Integer
```

Parameters

Return values

The number of member KSs in this KS Group.

Remarks

MemberName

Returns the name of member specified by index.

Example

```
Sub MemberName(nIndex As Integer) As String
```

Parameters

- Index - Index of the member KS

Return values

String value of the member KS name.

Remarks

ParseKSGroup

Parses the KS Group for its members.

Example

```
Sub ParseKSGroup()
```

Parameters

Return values

Remarks

ParseNewKSGroup

Parse to create a new KS Group.

Example

```
Sub ParseNewKSGroup(lpszName As String,lpszDesc As String,
Type As Integer)
```

Parameters

- lpszName - Name of the new KS Group
- lpszDesc - Description of the new KS Group
- Type - The type ID

  KSTYPE_REGULAR = 0

  KSTYPE_DISCOVERY = 1

  KSTYPE_INSTALL = 2

Return values

Remarks

RemoveMemberKS

Removes a member KS from the current KS Group.

Example

```
Sub RemoveMemberKS(Index As Integer)
```

Parameters

- Index - Index of member KS

Return values

Remarks

SearchMemberIndex

Returns the index to a member.

Example

```
Sub SearchMemberIndex(lpszName As String) As Integer
```

Parameters

- lpszName - Name of member KS

Return values

Index to the member KS.

Remarks

UpdateAll

Commit all changes and save KS Group.

Example

```
Sub UpdateAll()
```

Parameters

Return values

Remarks

Chapter 13

KSMatching object

This object represents the list of items that a Knowledge Script operates against. Items can be selected or deselected. By default, all matching items are selected.

This section covers the following topics:

- KSMatching properties
- KSMatching methods

KSMatching properties

The following properties are available for the KSMatching object.

Property	Description
Error	Indicates if the last operation resulted in an error.
GetCount	Returns the number of matching items
LastError	Indicates the error code of the last operation.
SelectionState	Determines if a matched item has been selected - by default all items are selected
SelectionStateMultiMachs	Gets/Sets a matched item to selected or unselected - by default all items are selected

Error

Indicates if the last operation resulted in an error.

Example

```
Property Error() As BOOL
```

Parameters

Return values

Boolean value indicating whether the last operation resulted in an error.

Remarks

Read-only attribute.

GetCount

Returns the number of matching items.

Example

```
Property GetCount As Integer
```

Parameters

Return values

The number of matching items for the computer at index = 0.

Remarks

Read-only attribute.

LastError

Indicates the error code of the last operation.

Example

```
Property LastError() As Integer
```

Parameters

Return values

The error number of the last operation.

Remarks

Read-only attribute.

SelectionState

Determines if a matched item has been selected - by default all items are selected.

Example

```
Property SelectionState(nIndex As Integer) As BOOL
```

Sample code

```
OjbectName = ‘‘CPU’’
BSelectState = FALSE
matchingCount = KSMatching.Count

If matchingCount > 0 Then
   For i = 0 to matchingCount - 1
   objNameStr = KSMatching.GetObjectName(i)
   If objNameStr = ObjectName Then
      KSMatching.SelectionState(i) = bSelectState
   End If
   Next
End If
```

Parameters

- nIndex - The index of the matching object

Return values

Boolean value indicating whether the matched object is selected for the computer at index = 0.

Remarks

SelectionStateMultiMachs

Gets/Sets a matched item to selected or unselected. By default all items are selected.

Example

```
Property SelectionStateMultiMachs(nIndex As Integer,long
machIndex) As BOOL
```

Sample code

```
machCount = oMatching.GetMachineCount()
for i = 0 to machCount - 1
      WScript.echo “Machine Name: “ & _
       oMatching.GetMachineName(i)
      nCount = oMatching.CountMultiMachs(i)
      WScript.echo “Matching Count: “&nCount
      for j = 0 to nCount - 1
         szObjectName = oMatching.GetObjectNameMultiMachs _
         &(j,i)
         bSelected = oMatching.SelectionStateMultiMachs _
         (j,i)
         WScript.echo “ObjectName: “szObjectName & “=” & _
         & bSelected
      Next
   Next
```

Parameters

- nIndex - Index of the matching object
- machIndex - Index of the computer

Return values

Boolean value indicating whether the matched object is selected.

Remarks

KSMatching methods

The following methods are available for the KSMatching object.

Method	Description
GetCountMultiMachs	Returns the number of matching items.
GetMachineCount.	Returns the number of machines available.
GetMachineName	Returns the machine name at the specified index.
GetMachineObjID	Returns the machine object ID at the specified index.

Method	Description
GetObjectIDMultiMachs	Gets the object ID of a matching item specified by an index.
GetObjectNameMultiMachs	Gets the object name of a matching item specified by an index.
SearchByObjID	Returns the index for matching item given a particular object ID.
SearchByObjIDMultiMachs	Returns the index for matching item given a particular object ID.

GetCountMultiMachs

Returns the number of matching items.

Example

```
Sub GetCountMultiMachs(machIndex As Long) As Integer
```

Parameters

- machIndex - Index of the computer

Return values

The number of matching objects for the specified computer.

Remarks

GetMachineCount.

Returns the number of machines available

Example

```
Sub GetMachineCount() As Long
```

Parameters

Return values

The number of computers.

Remarks

GetMachineName

Returns the machine name at the specified index.

Example

```
Sub GetMachineName(machIndex As Long) As String
```

Parameters

- machIndex - Index of the computer

Return values

String value of the computer name at the specified index.

Remarks

GetMachineObjID

Returns the machine object ID at the specified index.

Example

```
Sub GetMachineObjID(machIndex As Long) As 4 byte unsigned int
```

Parameters

- machIndex - Index of the computer

Return values

The computer's object ID at the specified index.

Remarks

GetObjectIDMultiMachs

Gets the object ID of a matching item specified by an index.

Example

```
Sub GetObjectIDMultiMachs(nIndex As Integer,machIndex As
Long) As Long
```

Sample code

```
machCount = oMatching.GetMachineCount()
for i = 0 to machCount - 1
      WScript.echo “Machine Name: “ & _
       oMatching.GetMachineName(i)
      nCount = oMatching.CountMultiMachs(i)
      WScript.echo “Matching Count: “&nCount
      for j = 0 to nCount - 1
         szObjectName = oMatching.GetObjectNameMultiMachs _
         &(j,i)
         bSelected = oMatching.SelectionStateMultiMachs _
         (j,i)
         WScript.echo “ObjectName: “szObjectName & “=” & _
         & bSelected
      Next

Next
```

Parameters

- nIndex - Index of the matching object
- machIndex - Index of the computer

Return values

The object ID of the matching object at the specified index for a specific computer.

Remarks

GetObjectNameMultiMachs

Gets the object name of a matching item specified by an index.

Example

```
Sub GetObjectNameMultiMachs(nIndex As Integer,machIndex As
Long) As String
```

Parameters

- nIndex - Index of the matching object
- machIndex - Index of the computer

Return values

The object name of the matching object at the specified index for a specific computer.

Remarks

SearchByObjID

Returns the index for matching item given a particular object ID.

Example

```
Sub SearchByObjID(strObjID As String) As Integer
```

Sample code

```
result = KSMatchingObject.SearchByObjeID("50")
Ir result > 0
      WScript.Echo "Object found!"
End if
```

Parameters

- strOjbID - String value of the object ID to find

Return values

The matching object index for a matching object for the computer at index = 0.

Remarks

SearchByObjIDMultiMachs

Returns the index for matching item given a particular object ID.

Example

```
Sub SearchByObjIDMultiMachs(strObjID As String,machIndex As
Long) As Integer
```

Parameters

- strObjID - String value of the object ID to find
- machIndex - Index of the computer

Return values

The matching object index.

Remarks

Chapter 14

KSParameters object

This object epresents the list of parameters for a given Knowledge Script. Use this object to enumerate and modify the parameters.

This section covers the following topics:

- KSParameters properties
- KSParameters methods

KSParameters properties

The following properties are available for the KSParameters object.

Property	Description
ComboBox	Indicates if the parameter should have an combobox: 0=none, 1=string.
Count	Returns the number of parameters in currently parsed KS.
Delimiters	Gets the delimiter of the specified parameter; parameter is specified by the index.
Description	Description of the specified parameter; parameter is specified by the index.
Error	Indicates if the last operation resulted in an error.
IsPassword	Indicates if the specified parameter is a password.
IsRequired	Indicates if the specified parameter is required.
ItemList	Returns a list of items in the combobox.
LastError	Indicates the error code of the last operation.
LegalCharacters	Indicates what characters are valid for the specified parameter.
MaxSize	Gets the maximum number of characters of the specified parameter.

Property	Description
MaxValue	Gets the maximum value allowed by the specified parameter.
Message	Gets the message of the specified parameter; parameter is specified by the index.
MinValue	Gets the minumum value allowed by the specified parameter.
Name	Gets the name of the specified parameter; parameter is specified by the index.
Type	Type of the specified parameter: 0=String, 1=Hex, 2=Number.
Units	Gets the units of the specified parameter; parameter is specified by the index.
Value	Set/Gets the value of the specified parameter. Parameter is specified by the index.

ComboBox

Indicates if the parameter should have an combobox.

Example

```
Property ComboBox(nIndex As Integer) As Integer
```

Parameters

- nIndex - index to the parameter

Return values

- 0 = None
- 1 = String

Remarks

Read-only attribute.

Count

Returns the number of parameters in currently parsed KS.

Example

```
Property Count() As Integer
```

Parameters

Return values

The number of parameters in the current KS.

Remarks

Read-only attribute.

Delimiters

Gets the delimiter of the specified parameter. The parameter is specified by the index.

Example

```
Property Delimiters(nIndex As Integer) As String
```

Parameters

- nIndex - Index to the parameter

Return values

The string value of the delimiter

Remarks

Read-only attribute.

Description

Description of the specified parameter.

Example

```
Property Description As String
```

Parameters

Return values

String value description of the parameter

Remarks

Read-only attribute.

Error

Indicates if the last operation resulted in an error.

Example

```
Property Error() As BOOL
```

Parameters

Return values

Boolean value indicating whether an error occurred during the last operation.

Remarks

A read-ony attribute.

IsPassword

Indicates if the specified parameter is a password.

Example

```
Property IsPassword(nIndex As Integer) As BOOL
```

Sample code

```
ParmName = “Severity” ‘Event severity
```

```
parmCount = KSParameters.Count
If parmCount > 0 Then
      nParameterIndex > -1 Then
         If KSParameters.IsPassword(nParameterIndex)
            Wscipt.Echo “This field is required!”
         End If
      End If
End If
```

Parameters

- nIndex - Index to the parameter

Return values

Boolean value indicating whether the parameter is a password.

Remarks

Read-only attribute.

IsRequired

Indicates if the specified parameter is required.

Example

```
Property IsRequired(nIndex As Integer) As BOOL
```

Parameters

nIndex - Index to the parameter

Return values

Boolean value indicating whether the parameter is required.

Remarks

Read-only attribute.

ItemList

Returns a list of items in the combobox.

Example

```
Property ItemList(nIndex As Integer) As String
```

Parameters

- nIndex - Index to the parameter

Return values

String value with the list of items in the combo box.

Remarks

Read-only attribute.

LastError

Indicates the error code of the last operation.

Example

```
Property LastError() As Integer
```

Parameters

Return values

The error number of the last operation.

Remarks

Read-only attribute.

LegalCharacters

Indicates what characters are valid for the specified parameter.

Example

```
Property LegalCharacters(nIndex As Integer) As String
```

Parameters

- nIndex - Index to the parameter

Return values

String value of characters that are valid for the parameter.

Remarks

Read-only attribute.

MaxSize

Gets the maximum number of characters of the specified parameter.

Example

```
Property MaxSize(nIndex As Integer) As Long
```

Parameters

- nIndex - Index to the parameter

Return values

The maximum number of characters allowed for the parameter.

Remarks

Read-only attribute.

MaxValue

Gets the maximum value allowed by the specified parameter.

Example

```
Property MaxValue(nIndex As Integer) As 8 byte real
```

Parameters

- nIndex - Index to the parameter

Return values

The maximum value allowed by the parameter.

Remarks

Read-only attribute.

Message

Gets the message of the specified parameter. The parameter is specified by the index.

Example

```
Property Message(nIndex As Integer) As String
```

Parameters

- nIndex - Index of the parameter

Return values

String value of the parameter message.

Remarks

Read-only attribute.

MinValue

Gets the minumum value allowed by the specified parameter.

Example

```
Property MinValue(nIndex As Integer) As 8 byte real
```

Parameters

- nIndex - Index of the parameter

Return values

The minimum value allowed by the parameter.

Remarks

Read-only attribute.

Name

Gets the name of the specified parameter. The parameter is specified by the index.

Example

```
Property Name(nIndex As Integer) As String
```

Sample code

```
parmCount = KSParameters.Count
For i = 0 to parmCount-1
   Wscript.Echo “parm” & i & “:” & KSParameters.Name(i)
Next
```

Parameters

- nIndex - Index of the parameter

Return values

String name of the parameter.

Remarks

Read-only attribute.

Type

Type of the specified parameter.

Example

```
Property Type(nIndex As Integer) As Integer
```

Parameters

- nIndex - Index of the parameter

Return values

- 0 = String
- 1 = Hexidecimal
- 2 = Number

Remarks

Read-only attribute.

Units

Gets the units of the specified parameter. The parameter is specified by the index.

Example

```
Property Units(nIndex As Integer) As String
```

Parameters

- nIndex - Index of the parameter

Return values

String value of the units.

Remarks

Read-only attribute.

Value

Set/Gets the value of the specified parameter. The parameter is specified by the index

Example

```
Property value(nIndex As Integer) As String
```

Sample code

```
ParmName = “Severity” ‘Event severity
NewValue = 15
parmCount = KSParameters.Count
If parmCount > 0 Then
      nParameterIndex = KSParameters.SearchByName _
      (ParmName)
      If nParameterIndex > -1 Then
         KSParameters.Value(nParameterIndex) = _
         CStr(newValue)
      End If
End If
```

Parameters

- nIndex - Index of the parameter

Return values

The value of the parameter.

Remarks

KSParameters methods

The following methods are available for the KSParameters object.

Method	Description
SearchByMessage	Searches the list of parameters by message and returns the index of that parameter if found.
SearchByName	Searches the list of parameters by name and returns the index of that parameter if found.

SearchByMessage

Searches the list of parameters by message and returns the index of that parameter if found.

Example

```
Sub SearchByMessage(lpszName As String) As Integer
```

Parameters

- lpszName - Message of the parameter

Return values

Index of the parameter.

Remarks

SearchByName

Searches the list of parameters by name and returns the index of that parameter if found.

Example

```
Sub SearchByName(lpszName As String) As Integer
```

Sample code

```
ParmName = Severity ‘Event severity
parmCount = KSParameters.Count
If parmCount > 0 Then
      nParameterIndex = KSParameters.SearchByName _
      (ParmName)
      If nParameterIndex > -1 Then
         Wscript.Echo “Parameter found at index: “ _
         &nParameterIndex
      End If
End If
```

Parameters

- lpszName - Message of the parameter

Return values

Index of the parameter.

Remarks

Chapter 15

KSProperties object

This object returns when a job or Knowledge Script is parsed. Use this object to get and set schedule information. This object also returns the KSParameters object and the KSActions object.

This section covers the following topics:

- KSProperties properties
- KSProperties methods

KSProperties properties

The following properties are available for the KSProperties object.

Property	Description
Actions	Returns a reference to the actions object.
AutoAck	Set/Get flag to acknowledge event automatically.
CollapsingInterval	Time interval for event collapsing.
DailyFrequency	Set/Get what kind of daily frequency is set (run once, every interval).
DailyFrequencyEvery	Returns the interval set for a daily schedule.
DailyFrequencyUnit	Returns the unit used to set a daily scheduling type.
DataCollectAverage	Set/Get flag for data average calculation.
DataCollectCfg	Set/Get flag to collect data.
DataCollectIter	Number of job iteration to collect data.
DataCollectOnEvent	Set/Get flag to collect data after event raised.
DataCollectOnEvent Stop	Set/Get flag for stop collecting data when event no longer exists.
DataUploadRDB	Set/Get flag to upload data to report repository.

Property	Description
EndDate	Set/Get end date of the scheduling.
EndTime	Set/Get end time of scheduling.
Error	Indicates if the last operation resulted in an error.
EventInterval	Number of occurrences before raising an event.
EventOccurrences	Number of consecutive occurrences before raising an event.
EveryNDays	Set/Get the number of days set for each interval of the schedule.
EveryNMonths	Set/Get the number of months set for each interval of the schedule.
EveryNTimes	Set/Get the number of iterations set for the scheduling.
EveryNWeeks	Set/Get the number of weeks set for each interval of the schedule.
IntervalUnit	Set/Get the units used to set the schedule interval.
IsActionKS	Indicates if this is an action KS.
IsDailyFreqSchedule	Determines if this is daily,weekly or monthly schedule.
IsIteration	Determines if schedule is iteration type.
IsMonthlyLastDay	Determines if this is last day of monthly schedule.
IsMonthlyNthDay	Determines if this is monthly-days scheduling.
IsMonthlyNthWeek	Determines if this is monthly-weeks scheduling.
IsWeeklyNthDay	Determines if this is weekly-days scheduling.
Iteration	Returns the interval of interation.
LastError	Indicates the error code of the last operation.
Matching	Returns a reference to the matching object.
MonthlyUnit	Set/Get the unit used to set the monthly schedule type.
Parameters	Returns a reference to the Parameters object.
RelativeInterval	Set/Get the relative interval used in a monthly schedule.
ScheduleMode	Determines what kind of scheduling is used (RUN ONCE, DAILY, etc).
StartDate	Set/Get the start date of the schedule.
StartNow	Determines if the schedule starts from now.

Property	Description
StartTime	Set/Get start time of the schedule.
StateChange	Set/Get flag of state change to raise event.
StateChangeSeverity	Minimum severity for event state change.
StopType	Set/Get how a job will stop (stop on, never stop, interation).
UseLastEvent	Set/Get flag to collapse duplicate events based on last event.
VersionCount	Get the number of versions available.
VersionID	Set/Get the current version id currently selected.

Actions

Returns a reference to the actions object.

Example

```
Property Actions() As Object
```

Parameters

Return values

A KSActions object reference for manipulating actions.

Remarks

AutoAck

Set/Get flag to automatically acknowledge event.

Example

```
Property AutoAck() As Long
```

Sample code

```
KSProperties.AutoAck = 0
```

Parameters

Return values

- 0 = Option off
- 1 = Option on

Remarks

CollapsingInterval

Time interval for event collapsing.

Example

```
Property CollapsingInterval() As Long
```

Parameters

Return values

Time interval for event collapsing in seconds. If set to 0, event collapsing is disabled.

Remarks

DailyFrequency

Set/Get what kind of daily frequency is set (run once, every interval).

Example

```
Property DailyFrequency() As Integer
```

Parameters

Return values

- 0 = Run once
- 1 = Run every interval

Remarks

DailyFrequencyEvery

Returns the interval set for a daily schedule.

Example

```
Property DailyFrequencyEvery() As Integer
```

Parameters

Return values

The interval set for the daily frequency.

Remarks

DailyFrequencyUnit

Returns the unit used to set a daily scheduling type.

Example

```
Property DailyFrequencyUnit() As Integer
```

Parameters

Return values

- 0 = Seconds
- 1 = Minutes
- 2 = Hours

Remarks

DataCollectAverage

Set/Get flag for data average calculation.

Example

```
Property DataCollectAverage() As Long
```

Sample code

```
KSProperties.DataCollectAverage = 0
```

Parameters

Return values

- 0 = Option off
- 1 = Option on

Remarks

DataCollectCfg

Set/Get flag to collect data

Example

```
Property DataCollectCfg() As Long
```

Parameters

Return values

0 = Collect data points only

1 = Collect data points and data details

Remarks

DataCollectIter

Number of job iterations to collect data.

Example

```
Property DataCollectIter() As Long
```

Parameters

Return values

Number of job interations during which data is collected.

Remarks

DataCollectOnEvent

Set/Get flag to collect data after an event is raised.

Example

```
Property DataCollectOnEvent() As Long
```

Parameters

Return values

- 0 = Option off
- 1 = Option on

Remarks

DataCollectOnEventStop

Set/Get flag for stopping data collection when event condition no longer exists.

Example

```
Property DataCollectOnEventStop() As Long
```

Parameters

Return values

- 0 = Option off
- 1 = Option on

Remarks

DataUploadRDB

Set/Get flag to upload data to report repository.

Example

```
Property DataUploadRDB() As Long
```

Parameters

Return values

- 0 = Option off
- 1 = Option on

Remarks

EndDate

Set/Get end date of the scheduling.

Example

```
Property EndDate() As Long
```

Parameters

Return values

The end date of the scheduling in AppManager date format.

Remarks

EndTime

Set/Get end time of scheduling.

Example

```
Property EndTime() As Long
```

Parameters

Return values

The end time of the scheduling in AppManager time format.

Remarks

Error

Indicates if the last operation resulted in an error.

Example

```
Property Error() As BOOL
```

Parameters

Return values

- True = Error occurred
- False = No error

Remarks

EventInterval

Number of occurrences before raising an event.

Example

```
Property EventInterval() As Long
```

Sample code

KSProperties.EventInterval = 5

Parameters

Return values

Number of job iterations to look for event condition.

Remarks

EventOccurrences

Number of consecutive occurrences before raising an event.

Example

```
Property EventOccurrences() As Long
```

Parameters

Return values

Number of consecutive occurrences before raising an event.

Remarks

EveryNDays

Set/Get the number of days set for each interval of the schedule.

Example

```
Property EveryNDays() As Long
```

Parameters

Return values

Number of days set for each interval of the schedule.

Remarks

EveryNMonths

Set/Get the number of months set for each interval of the schedule.

Example

```
Property EveryNMonths() As Integer
```

Parameters

Return values

Number of months set for each interval of the schedule.

Remarks

EveryNTimes

Set/Get the number of iterations set for the scheduling.

Example

```
Property EveryNTimes() As Integer
```

Parameters

Return values

The number of iterations set for the scheduling.

Remarks

EveryNWeeks

Set/Get the number of weeks set for each interval of the schedule.

Example

```
Property EveryNWeeks() As Integer
```

Parameters

Return values

The number of weeks set for each interval of the schedule.

Remarks

IntervalUnit

Set/Get the units used to set the schedule interval.

Example

```
Property IntervalUnit() As Integer
```

Parameters

Return values

- 0 = Seconds
- 1 = Minutes
- 2 = Hours

Remarks

IsActionKS

Indicates if this is an action KS.

Example

```
Property IsActionKS() As BOOL
```

Parameters

Return values

Boolean value indicating if the KS is an Action KS.

Remarks

Read-only attribute.

IsDailyFreqSchedule

Determines if this is daily, weekly or monthly schedule.

Example

```
Property IsDailyFreqSchedule() As BOOL
```

Parameters

Return values

Boolean value indicating whether the schedule is daily, weekly, or monthly.

Remarks

Read-only attribute.

IsIteration

Determines if schedule is iteration-type.

Example

```
Property IsIteration() As BOOL
```

Parameters

Return values

Boolean value indicating if the schedule is iteration-type.

Remarks

Read-only attribute.

IsMonthlyLastDay

Determines if this is the last day of a monthly schedule.

Example

```
Property IsMonthlyLastDay() As BOOL
```

Parameters

Return values

Boolean value indicating if this is the last day of a monthly schedule.

Remarks

Read-only attribute.

IsMonthlyNthDay

Determines if this is monthly/days scheduling.

Example

```
Property IsMonthlyNthDay As BOOL
```

Parameters

Return values

Boolean value indicating if this schedule is monthly/days.

Remarks

Read-only attribute.

IsMonthlyNthWeek

Determines if this is monthly/weeks scheduling.

Example

```
Property IsMonthlyNthWeek As BOOL
```

Parameters

Return values

Boolean value indicating if this schedule is monthly/weeks.

Remarks

Read-only attribute.

IsWeeklyNthDay

Determines if this is weekly/days scheduling.

Example

```
Property IsWeeklyNthDay As BOOL
```

Parameters

Return values

Boolean value indicating if this schedule is weekly/days.

Remarks

Read-only attribute.

Iteration

Returns the interval of interation.

Example

```
Property Iteration() As BOOL
```

Parameters

Return values

The interval of iteration.

Remarks

LastError

Indicates the error code of the last operation.

Example

```
Property LastError() As Integer
```

Parameters

Return values

The error code of the last operation.

Remarks

Read-only attribute.

Matching

Returns a reference to the matching object.

Example

```
Property Matching() As Object
```

Parameters

Return values

A KSMatching object reference used to specify objects.

Remarks

MonthlyUnit

Set/Get the unit used to set the monthly schedule type.

Example

```
Property MonthlyUnit() As Integer
```

Parameters

Return values

- 0 = MONTHLY_SUB_SUNDAY
- 1 = MONTHLY_SUB_MONDAY
- 2 = MONTHLY_SUB_TUESDAY
- 3 = MONTHLY_SUB_WEDNESDAY
- 4 = MONTHLY_SUB_THURSDAY
- 5 = MONTHLY_SUB_FRIDAY
- 6 = MONTHLY_SUB_SATURDAY
- 7 = MONTHLY_SUB_DAY
- 8 = MONTHLY_SUB_WEEKDAY
- 9 = MONTHLY_SUB_WEEKEND

Remarks

Parameters

Returns a reference to the Parameters object.

Example

```
Property Parameters() As Object
```

Parameters

Return values

A KSParameters object reference used to get and set specific parameters.

Remarks

RelativeInterval

Set/Get the relative interval used in a monthly schedule.

Example

```
Property RelativeInterval() As Long
```

Parameters

Return values

The relative interval used in a monthly schedule.

Remarks

ScheduleMode

Determines what kind of scheduling is used (RUN ONCE, DAILY, etc.).

Example

```
Property ScheduleMode() As Integer
```

Parameters

- 0 = SCHEDULE_RUNONCE
- 1 = SCHEDULE_INTERVAL
- 2 = SCHEDULE_XTIMES
- 3 = SCHEDULE_DAILY
- 4 = SCHEDULE_WEEKLY
- 5 = SCHEDULE_MONTHLY
- 6 = SCHEDULE_ASYNC

Return values

Remarks

StartDate

Set/Get the start date of the schedule.

Example

```
Property StartDate() As Long
```

Parameters

Return values

Start date of the schedule in AppManager date format.

Remarks

StartNow

Determines if the schedule starts from now.

Example

```
Property StartNow() As BOOL
```

Parameters

Return values

Boolean value indicating if the schedule starts from now.

Remarks

StartTime

Set/Get start time of the schedule.

Example

```
Property StartTime() As Long
```

Parameters

Return values

Start time of the schedule in AppManger time format.

Remarks

StateChange

Set/Get flag of state change to raise event.

Example

```
Property StateChange() As Long
```

Parameters

Return values

- 0 = Option off
- 1 = Option on

Remarks

StateChangeSeverity

Minimum severity for event state change.

Example

```
Property StateChangeSeverity() As Long
```

Sample code

KSProperties.StateChangeSeverity = 20

Parameters

Return values

Severity of the new event indicating the original event condition no longer exists.

Remarks

StopType

Set/Get how a job will stop (stop on, never stop, interation).

Example

```
Property StopType() As Integer
```

Parameters

Return values

- 0 = STOP_TIME_NEVER
- 1 = STOP_TIME_ON
- 2 = STOP_TIME_ITERATION

Remarks

UseLastEvent

Set/Get flag to collapse duplicate events based on last event.

Example

```
Property UseLastEvent() As Long
```

Parameters

Return values

- 0 = Base on initial event
- 1 = Base on last event

Remarks

VersionCount

Get the number of versions available.

Example

```
Property VersionCount() As Integer
```

Parameters

Return values

Number of versions available for this KS.

Remarks

Read-only attribute.

VersionID

Set/Get the current version ID currently selected.

Example

```
Property VersionID() As 4 byte unsigned int
```

Parameters

- 1 = KS_VERSION_3
- 2 = KS_VERSION_4

Return values

Remarks

KSProperties methods

The following methods are available for the KSProperties object.

Method	Description
IsTypeAllowed	Checks to see if the KS allows for the specified schedule type.
SetAsyncScheduling	Sets up a knowledge script to be asychronously.
SetDailyScheduling	Sets up a daily schedule for a knowledge script.
SetIntervalScheduling	Sets up a interval be run on specific intervals.
SetMonthlyScheduling	Sets up a monthly schedule for a knowledge script.
SetRunOnceScheduling	Sets up a knowledge script to be run once.
SetWeeklyScheduling	Sets up KS to be scheduled weekly.
SetXTimesScheduling	Sets up knowledge script to be run every X times.
TranslateFromNetIQDate	Translates an AppManager date to an OLE date value.
TranslateFromNetIQTime	Translates an AppManager time to an OLE time value.
TranslateToNetIQDate	Translates an OLE date value to an AppManager date value.
TranslateToNetIQTime	Translates an OLE time value to an AppManager time value.
ValidateScheduleTime	Checks to see that the date/times are correct depending on the scheduling mode; returns an error code.
VersionIDByIndex	Get the version id of a specified index.

IsTypeAllowed

Checks to see if the KS allows for the specified schedule type.

Example

```
Sub IsTypeAllowed(type as Long) As Boolean
```

Parameters

- type - The scheduling type

  1 = TYPE_ONE_TIME

  2 = TYPE_ON_DEMAND

  4 = TYPE_RECURRING_DAILY

  8 = TYPE_RECURRING_ WEEKLY

  16 = TYPE_RECURRING_MONTHLY1

  32 = TYPE_RECURRING_MONTHLY2

  64 = TYPE_AUTO_START ‘same as Async

  128 = TYPEP_INTERVAL_ITERATION

Return values

TRUE if type is allowed; otherwise FALSE.

Remarks

SetAsyncScheduling

Sets up a knowledge script to be asychronously scheduled.

Example

```
Sub SetAsyncScheduling()
```

Parameters

Return values

Remarks

SetDailyScheduling

Sets up a daily schedule for a knowledge script.

Example

```
Sub SetDailyScheduling(bStartNow As BOOL,nStopType As
Integer,nDailyFreq As Integer,nDailyFreqUnit As
Integer,nEveryXDays As Long,nEveryXFreq As Integer,StartDate
As Long,EndDate As Long,Starttimeofday As Long,Endtimeofday
As Long)
```

Parameters

- bStartNow - If set to True, run job immediately
- nStopType - When to stop the job
- nDailyFreq - The type of daily frequency
- nDailyFreqUnit - The daily units to run the job
- nEveryXDays - Run the job every X number of days
- nEveryXFreq - The number of intervals for the daily unit
- StartDate - The date to start the job in AppMangaer date format
- EndDate - The date to end the job in AppManager date format
- Starttimeofday - The time to start the job in AppManager time format
- Endtimeofday - The time to end the job in AppManager time format

Return values

Remarks

SetIntervalScheduling

Sets up an job to run on specific intervals.

Example

```
Sub SetIntervalScheduling(bStartNow As BOOL,nStopType As
Integer,nIntervalUnit As Integer,nEveryXInterval As
Integer,StartDate As Long,Starttimeofday As Long,EndDate As
Long,Endtimeofday As Long,Iteration As Long)
```

Sample code

```
bStartNow = False
startDate  = KSProperties.TranslateToNetIQDate(1999,11,13)
startTime = KSProperties.TranslateToNetIQTime(13,30,00)
stopType = STOP_TIME_NEVER

endDate = KSProperties.TranslateToNetIQDate(1999,11,15)
endTime = KSProperties.TranslateToNetIQTime(13,45,00)

everyXInterval = 5
intervalUnit = INTERVAL_MINUTES
Iteration = 0

KS.Properties.SetIntervalScheduling bStartNow, stopType,
intervalUnit, everyXInterval, StartDate, StartTime, endDate,
endTime, Iteration
```

Parameters

- bStartNow - If set to True, run job immediately
- nStopType - When to stop the job
- nIntervalUnit - Indicate the type of units to use
- nEveryXInterval - Set the number of intervals
- StartDate - Date to start the job in AppManager date format
- Starttimeofday - Time to start the job in AppManager time format

- EndDate - Date to end the job in AppManager date format
- Endtimeofday - Time to end the job in AppManager time format
- Iteration - This parameter is obsolete; set to 0

Return values

Remarks

SetMonthlyScheduling

Sets up a monthly schedule for a KS.

Example

```
Sub SetMonthlyScheduling(bStartNow As BOOL,nStopType As
Integer,nMonthlyUnit As Integer,nMonthDaysBits As
Long,nMonthWeeks As Integer,nEveryXMonths As
Integer,nEveryXFreq As Integer,StartDate As Long,EndDate As
Long,Starttimeofday As Long,Endtimeofday As Long)
```

Sample code

```
bStartNow = False
stopType = STOP_TIME_ON
startDate = KSProperties.TranslateToNetIQDate(1999,11,13)
endDate = KSProperties.TranslateToNetIQDate(1999,11,15)

everyXMonths = 2
dailyFreq = DAILY_FREQ_EVERY

everyXFreq = 8
dailyFreqUnit = INTERVAL_SECONDS

monthDaysBits = CLng(MONTHLY_DAY_13 + MONTHLY_DAY_15 +
MONTHLY_DAY_19 + MONTLY_DAY_31)

monthlyUnit = MONTHLY_SUB_DAY

startTime = KSProperties.TranslateToNetIQTime(11,30,00)
endTime = KSProperties.TranslateToNetIQTime(11,45,00)

KSProperties.SetMonthlyScheduling bStartNow, stopType,
MonthlyUnit, dailyFreq, dailyFreqUnit, monthDaysBits,
```

```
monthWeeks, everyXMonths, everyXFreq, startDate, endDate,
StartTime, EndTime, KSProperties.StopType = stopType
```

Parameters

- bStartNow - If set to True, run job immediately
- nStopType - When to stop the job
- nMonthlyUnit - The units of the days in the month
- nDailyFreq - The type of daily frequency
- nDailyFreqUnit - The daily units to run the job
- nMonthDaysBits - Depending on the monthly units, select days of the month
- nMonthWeeks - Depending on the monthly units, select the weeks of the month
- nEveryXMonths - Run job every X number of months
- nEveryXFreq - The number of intervals for the daily unit
- StartDate - The date to start the job in AppManger date format
- EndDate - The date to end the job in AppManager date format
- Starttimeofday - The time to start the job in AppManager time format
- Endtimeofday - The time to end the job in AppManager time format

Return values

Remarks

SetRunOnceScheduling

Sets up a knowledge script to be run once.

Example

```
Sub SetRunOnceScheduling(bStartNow As BOOL,StartDate As
Long,Starttimeofday As Long)
```

Sample code

```
bStartNow = False
startDate = KSProperties.TranslateToNetIQDate(2001,11,24)
startTime = KSProperties.TranslateToNetIQTime(13,30,00)
KS.Properties.SetRunOnceScheduling bStartNow, startDate,
startTime
```

Parameters

- bStartNow - If set to True, run job immediately
- StartDate - Date to start the job in AppManager date format
- Starttimeofday - Time to start the job in AppManager time format

Return values

Remarks

SetWeeklyScheduling

Sets up KS to be scheduled weekly.

Example

```
Sub SetWeeklyScheduling(bStartNow As BOOL,nStopType As
Integer,nDailyFreq As Integer,nDailyFreqUnit As
Integer,nWeekDaysBits As Long,nEveryXWeeks As
Integer,nEveryXFreq As Integer,StartDate As Long,EndDate As
Long,Starttimeofday As Long,Endtimeofday As Long)
```

Parameters

- bStartNow - If set to True, run job immediately
- nStopType - When to stop the job
- nDailyFreq - The type of daily frequency
- nDailyFreqUnit - The daily units to run the job
- nWeekDaysBits - Select days of the week
- nEveryXWeeks - Run the job every X number of weeks
- nEveryXFreq - The number of intevals for the daily unit

- StartDate - Date to start the job in AppManager date format
- EndDate - Date to end the job in AppManager date format
- Starttimeofday - Time to start the job in AppManager time format
- Endtimeofday - Time to end the job in AppManger time format

Return values

Remarks

SetXTimesScheduling

Sets up knowledge script to be run every X times

Example

```
Sub SetXTimesScheduling(bStartNow As BOOL,nStopType As
Integer,nIntervalUnit As Integer,nEveryXInterval As
Integer,StartDate As Long,Starttimeofday As Long,EndDate As
Long,Endtimeofday As Long,Iteration As Long)
```

Parameters

- bStartNow - If set to True, run job immediately
- nStopType - When to stop the job
- nIntervalUnit - Indicate the type of units to use
- nEveryXInterval - Number of intervals
- StartDate - Date to start the job in AppManager date format
- EndDate - Date to end the job in AppManager date format
- Starttimeofday - Time to start the job in AppManager time format
- Endtimeofday - Time to end the job in AppManager time format
- Iteration - Number of times to run the job

Return values

Remarks

TranslateFromNetIQDate

Translates an AppManager date to an OLE date value.

Example

```
Sub TranslateFromNetIQDate(dtParm As Long) As Date
```

Parameters

- dtParm - A date in AppManager format

Return values

A date in standard OLE format.

Remarks

TranslateFromNetIQTime

Translates an AppManager time to an OLE time value.

Example

```
Sub TranslateFromNetIQTime(dtParm As Long) As Date
```

Sample code

```
startTime =
KSProperties.TranslateFromNetIQTime(KSProperties.StartTime)
WScript.Echo “Start Time: “ & startTime
```

Parameters

- dtParm - A time in AppManager format.

Return values

A time in standard OLE format

Remarks

TranslateToNetIQDate

Translates an OLE date value to an AppManager date value.

Example

```
Sub TranslateToNetIQDate(year As Integer,month As Integer,day
As Integer) As Long
```

Parameters

- year - The year
- month - The numerical month
- day - The numerical day of the month

Return values

A date in AppManager format.

Remarks

TranslateToNetIQTime

Translates an OLE time value to an AppManager time value.

Example

```
Sub TranslateToNetIQTime(hour As Integer,minute As
Integer,second As Integer) As Long
```

Parameters

- hour - The hour
- minute - The number of minutes
- second - The number of seconds

Return values

A time in AppManager format.

Remarks

ValidateScheduleTime

Checks to see that the date/times are correct depending on the scheduling mode; returns an error code.

Example

```
Sub ValidateScheduleTime() As Integer
```

Parameters

Return values

- 0 = SUCCESS_SCHEDULE
- 1000 = ERROR_SCHEDULE_WEEKLY_NEED_DAY
- 1001 = ERROR_SCHEDULE_MONTHLY_NEED_DAY
- 1002 = ERROR_SCHEDULE_MONTHLY_NEED_WEEK
- 1003 = ERROR_SCHEDULE_STARTDATE_ET_CURDATE
- 1004 = ERROR_SCHEDULE_STARTTIME_ET_CURTIME
- 1005 = ERROR_SCHEDULE_STARTDATE_LT_STOPDATE
- 1006 = ERROR_SCHEDULE_STARTTIME_LE_STOPTIME
- 1007 = ERROR_SCHEDULE_DAILY_STARTDATE_EQ_STOPDATE
- 1008 = ERROR_SCHEDULE_DAILY_STARTTIME_EQ_STOPTIME

Remarks

VersionIDByIndex

Get the version ID of a specified index.

Example

```
Sub VersionIDByIndex(nIndex As Integer) As 4 byte unsigned int
```

Parameters

- nIndex - The index for the version

Return values

The version ID by index.

Remarks

Read-only attribute.

Chapter 16

Machines object

This object represents the list of known computers.

This section covers the following topics:

- Machines properties
- Machines methods

Machines properties

The following properties are available for the Machines object.

Property	Description
AutoRefresh	A flag indicating whether the recordset will be requeried after any operation which could affect the recordset (useful for batch operations).
BOF	Indicates that the current record position is before the first record.
Count	Returns the number of records in a recordset.
EOF	Indicates the current record position is after the last record.
Error	Indicates if the last operation resulted in an error.
Fields	Returns an instance of a field's object.
LastError	Indicates the error code of the last operation.
MCVersion	Returns the mc version of the machine.
NewServerType	Get/Set the machine's type when adding a new server.
NTAutoDiscovery	Discover NT resources automatically when add a computer.
Properties	Returns a reference to the MachProperties object for the machine specified by the current record position.

Property	Description
SortIndex	Specifies which column in a recordset should be sorted on.
View	Sets/Gets the view ID which will be used in view-specific objects.

AutoRefresh

A flag indicating whether the recordset will be requeried after any operation which could affect the recordset (useful for batch operations).

Example

```
Property AutoRefresh() As BOOL
```

Parameters

Return values

A boolean expression that determines whether to automatically requery the recordset.

Remarks

BOF

Indicates that the current record position is before the first record.

Example

```
Property BOF() As BOOL
```

Parameters

Return values

A boolean expression.

Remarks

Read-only attribute.

Count

Returns the number of records in a recordset.

Example

```
Property Count() As Long
```

Parameters

Return values

The number of records in a recordset.

Remarks

Read-only attribute.

EOF

Indicates the current record position is after the last record.

Example

```
Property EOF() As BOOL
```

Parameters

Return values

A boolean expression.

Remarks

Read-only attribute.

Error

Indicates if the last operation resulted in an error.

Example

```
Property Error() As BOOL
```

Parameters

Return values

A boolean expression to indicate if an error occurred.

Remarks

Read-only attribute.

Fields

Returns an instance of a field's object.

Example

```
Property Fields() As Object
```

Parameters

Return values

An instance of a field's object.

Remarks

Read-only attribute.

LastError

Indicates the error code of the last operation.

Example

```
Property LastError() As Integer
```

Parameters

Return values

The error number of the last operation.

Remarks

Read-only attribute.

MCVersion

Returns the mc version of the machine.

Example

```
Property MCVersion() As 4 byte unsigned int
```

Parameters

Return values

- 1 = mc version 3
- 2 = mc version 4

Remarks

Read-only attribute.

NewServerType

Get/Set the machine's type when adding a new server. The default is NT.

Example

```
Property NewServerType() As Integer
```

Parameters

Return values

- 0 = NT Server

- 1 = UNIX server

Remarks

NTAutoDiscovery

Discover NT resources automatically when adding a computer.

Example

```
Property NTAutoDiscovery() As BOOL
```

Parameters

Return values

Boolean value indicating whether to automatically discover NT resources.

Remarks

Properties

Returns a reference to the MachProperties object for the machine specified by the current record position.

Example

```
Property Properties() As Object
```

Parameters

Return values

A MachProperties object reference.

Remarks

SortIndex

Specifies which column in a recordset should be sorted on.

Example

```
Property SortIndex() As Integer
```

Parameters

Return values

The current sort index number.

Remarks

View

Sets/Gets the view ID which will be used in view-specific objects.

Example

```
Property View() As Integer
```

Parameters

Return values

The number ID of the view to limit by.

Remarks

Machines methods

The following methods are available for the Machines object.

Method	Description
AddServer	Adds a computer to the known list of computers.
AddServer2	Adds a computer to the known list of computers - allows setting check flags.
AddServerToSubView	Adds a new or existing server to a master view based subview.
CreateServerGroup	Creates a server group in the treeview.
CreateServerGroupEx	Creates a server group in the treeview in specified view.
DeleteServer	Deletes a computer - will fail if computer has any outstanding jobs or events.
DeleteServerEx	Delete a machine from a server group in the treeview in specified view.
GetField	Retrieves the data for a particular column in a record.
JumpAbsolute	Repositions the record pointer to the specified record number.
JumpRelative	Repositions the record pointer x number of spaces from where it was last.
MoveFirst	Moves the record position to the first record in the recordset.
MoveLast	Moves the record position to the last record in the recordset.
MoveNext	Moves the record position to the record after the current record.
MovePrevious	Moves the record position to the record before the current record.
MoveServer	Move a machine to a server group.
MoveServerEx	Move a machine to a server group in the treeview in specified view.
Refresh	Refreshes the current recordset.

Method	Description
Search	A non case-sensitive search on a particular column.
SetFilter	Applies a filter to the current recordset.

AddServer

Adds a computer to the known list of computers.

Example

```
Sub AddServer(strServerName As String,parentObjID As Long) As
Long
```

Parameters

- strServerName - Name of the computer
- parentObjID - The computer's parent object ID

Return values

The new computer's object ID. 0 is returned if the computer is not added.

Remarks

AddServer2

Adds a computer to the known list of computers, and allows setting check flags.

Example

```
Sub AddServer2(strServerName As String,parentObjID As
Long,checkFlags As Long) As Long
```

Parameters

- strServerName - Name of the computer
- parentObjID - The computer's parent object ID
- checkFlags

CREATESERVER_CHECK_NONE 0x00000000

CREATESERVER_CHECK_ALL 0xFFFFFFFE

CREATESERVER_BASIC 0x00000002

Return valus

The new computer's object ID. 0 is returned if the computer is not added.

Remarks

AddServerToSubView

Adds a new or existing server to a Master view-based subview.

Example

```
Sub AddServerToSubView(strServerName As String,parentObjID As
Long,viewID As Long) As Long
```

Parameters

- strServerName - Name of the computer
- parentObjID - The computer's parent object ID
- viewID - The computer's view ID

Return values

0 if successful; otherwise a negative value.

Remarks

CreateServerGroup

Creates a server group in the treeview.

Example

```
Sub CreateServerGroup(strGroupName As String,parentObjID As
Long) As Long
```

Parameters

- strGroupName - Name of the server group
- parentObjID - The server group's parent object ID

Return values

The server group object ID. 0 and negative values if the method fails.

Remarks

CreateServerGroupEx

Creates a server group in the treeview in specified view.

Example

```
Sub CreateServerGroupEx(strGroupName As String,parentObjID As
Long,viewID As Long) As Long
```

Parameters

- strGroupName - Name of the server group
- parentObjID - The server group's parent object ID
- viewID - The server group's view ID

Return values

The server group object ID. 0 and negative values if the method fails.

Remarks

DeleteServer

Deletes a computer. Fails if computer has any outstanding jobs or events.

Example

```
Sub DeleteServer(machGroupID As Long)
```

Parameters

- machGroupID - Object ID of the computer

Return values

Remarks

DeleteServerEx

Delete a machine from a server group in the treeview in specified view.

Example

```
Sub DeleteServerEx(machObjID As Long,viewID As Long)
```

Parameters

- machObjID - The computer's object ID
- viewID - The computer's view ID

Return values

Remarks

GetField

Retrieves the data for a particular column in a record.

Example

```
Sub GetField(column As Integer) As String
```

Parameters

- column - The column of the desired field

Return values

The data for a particular column in a record.

Remarks

JumpAbsolute

Repositions the record pointer to the specified record number.

Example

```
Sub JumpAbsolute(offset As Long)
```

Parameters

Return values

Remarks

JumpRelative

Repositions the record pointer x number of spaces from where it was last.

Example

```
Sub JumpAbsolute(offset As Long)
```

Parameters

Return values

Remarks

MoveFirst

Moves the record position to the first record in the recordset.

Example

```
Sub MoveFirst()
```

Parameters

Return values

Remarks

MoveLast

Moves the record position to the last record in the recordset.

Example

```
Sub MoveLast()
```

Parameters

Return values

Remarks

MoveNext

Moves the record position to the record after the current record.

Example

```
Sub MoveNext()
```

Parameters

Return values

Remarks

MovePrevious

Moves the record position to the record before the current record.

Example

```
Sub MovePrevious()
```

Parameters

Return values

Remarks

MoveServer

Move a machine to a server group.

Example

```
Sub MoveServer(objid As Long,parentObjID As Long) As Long
```

Parameters

- objid - The computer's object ID
- parentObjID - The computer's parent object ID

Return values

- 0 = success
- -1 = failure

Remarks

MoveServerEx

Move a machine to a server group in the treeview in specified view.

Example

```
Sub MoveServerEx(objid As Long,oldParentObjID As
Long,newParentObjID As Long,viewID As Long) As Long
```

Parameters

- objid - The computer's object ID
- oldParentObjID - The computer's current parent object ID
- newParentObjID - The computer's new parent object ID

- viewID - The computer's view ID

Return values

- 0 = success
- -1 = failure

Remarks

Refresh

Refreshes the current recordset.

Example

```
Sub Refresh()
```

Parameters

Return values

Remarks

Search

A non case-sensitive search on a particular column.

Example

```
Sub Search(column As Integer,searchStr As String)
```

Parameters

- column - Number indicating which column to search on
- searchStr - The string to search for

Return values

Remarks

SetFilter

Applies a filter to the current recordset

Example

```
Sub SetFilter(filterType As Integer,parm1 As String,parm2 As
String)
```

Parameters

- filterType - ID of the filter

  0 = //|No Filter

  1 = //obj id|Filter by Object ID

  2 = //machine|Filter by Machine

  3 = //view|Filter by View

  4 = //obj id and view|Filter by Object ID and View

  5 = //machine and view|Filter by Machine and View

  6 = //server group ID|Filter by Server Group ID

- parm 1 - Value of parameter 1 if the filter requires one or more parameters, otherwise pass in an empty string
- parm 2 - Value of parameter 2 if the filter requires a second parameter, otherwise pass in an empty string

Return values

Remarks

Chapter 17

Misc object

The Misc object provides miscellaneious utility functions.

This section covers the following topics:

- Misc properties
- Misc methods

Misc properties

The following properties are available for the Misc object.

Property	Description
Error	Indicates if the last operation resulted in an error.
LastError	Indicates the error code of the last operation.

Error

Indicates if the last operation resulted in an error.

Example

```
Property Error() As BOOL
```

Parameters

Return values

Boolean value indicating whether the last operation resulted in an error.

Remarks

LastError

Indicates the error code of the last operation.

Example

```
Property LastError() As Integer
```

Parameters

Return values

The error code of the last operation.

Remarks

Read-only attribute.

Misc methods

The following methods are available for the Misc object

Property	Description
DATEtoUTC	Converts an OLE date/time to a UTC value.
GetActiveTimeBias	Returns the active time bias with DST adjustment.
GetBias	Returns the bias without DST adjustment.
GetDST	Returns 1 if DST is on, and 0 otherwise.
GetDstBias	Returns the DST bias.
GetSummerTime	Returns 1 if today is from March 30 to October 26, and 0 otherwise.
GetTimeZone	Returns an offset of the GMT.
UTCtoDate	Converts a UTC value to an OLE date/time format.

DATEtoUTC

Converts an OLE date/time to a UTC value.

Example

```
Sub DATEtoUTC(dt As Date) As Long
```

Parameters

- dt - A date in OLE format

Return values

A date/time in UTC format.

Remarks

GetActiveTimeBias

Returns the active time bias with DST adjustment.

Example

```
Sub GetActiveTimeBias() As Long
```

Parameters

Return values

The active time bias with DST adjustment.

Remarks

GetBias

Returns the bias without DST adjustment.

Example

```
Sub GetBias() As Long
```

Parameters

Return values

The bias without a DST adjustment.

Remarks

GetDST

Returns 1 if DST is on, and 0 otherwise.

Example

```
Sub GetDST() As Long
```

Parameters

Return values

1 if DST is on; 0 otherwise.

Remarks

GetDstBias

Returns the DST bias.

Example

```
Sub GetDstBias() As Long
```

Parameters

Return values

The DST bias.

Remarks

GetSummerTime

Returns 1 if today is from March 30 to October 26, and 0 otherwise.

Example

```
Sub GetSummerTime() As Long
```

Parameters

Return values

1 if today is March 30 to October 26; 0 otherwise.

Remarks

GetTimeZone

Returns an offset of GMT.

Example

```
Sub GetTimeZone() As Long
```

Parameters

Return values

The offset of GMT.

Remarks

UTCtoDate

Converts a UTC value to an OLE date/time format.

Example

```
Sub UTCtoDATE(utcValue As 8 byte real) As Date
```

Parameters

- utcValue - Date/time in UTC format

Return values

A date/time in OLE format.

Remarks

Chapter 18

Monitor object

This object represents the list of computers with open events that fall within a specified event severity range.

This section covers the following topics:

- Monitor properties
- Monitor methods

Monitor properties

The following properties are available for the Monitor object.

Property	Description
AutoRefresh	A flag indicating whether the recordset will be requeried after any operation which could affect the recordset (useful for batch operations).
BOF	Indicates that the current record position is before the first record.
Count	Returns the number of records in a recordset.
EOF	Indicates the current record position is after the last record.
Error	Indicates if the last operation resulted in an error.
Fields	Returns an instance of a field's object.
LastError	Indicates the error code of the last operation.
SortIndex	Specifies which column in a recordset should be sorted on.
View	Sets/Gets the view ID which will be used in view-specific objects

AutoRefresh

A flag indicating whether the recordset will be requeried after any operation which could affect the recordset (useful for batch operations).

Example

```
Property AutoRefresh() As BOOL
```

Parameters

Return values

A boolean expression that determines whether to automatically requery the recordset.

Remarks

BOF

Indicates that the current record position is before the first record.

Example

```
Property BOF() As BOOL
```

Parameters

Return values

A boolean expression.

Remarks

Read-only attribute.

Count

Returns the number of records in a recordset.

Example

```
Property Count() As Long
```

Parameters

Return values

The number of records in a recordset.

Remarks

Read-only attribute.

EOF

Indicates the current record position is after the last record.

Example

```
Property EOF() As BOOL
```

Parameters

Return values

A boolean expression.

Remarks

Read-only attribute.

Error

Indicates if the last operation resulted in an error.

Example

```
Property Error() As BOOL
```

Parameters

Return values

A boolean expression to indicate if an error occurred.

Remarks

Read-only attribute.

Fields

Returns an instance of a field's object.

Example

```
Property Fields() As Object
```

Parameters

Return values

An instance of a field's object.

Remarks

Read-only attribute.

LastError

Indicates the error code of the last operation.

Example

```
Property LastError() As Integer
```

Parameters

Return values

The error number of the last operation.

Remarks

Read-only attribute.

SortIndex

Specifies which column in a recordset should be sorted on.

Example

```
Property SortIndex() As Integer
```

Parameters

Return values

The current sort index number.

Remarks

View

Sets/Gets the view ID which will be used in view-specific objects.

Example

```
Property View() As Integer
```

Parameters

Return values

The number ID of the view to limit by.

Remarks

Monitor methods

The following methods are available for the Monitor object.

Method	Description
GetField	Retrieves the data for a particular column in a record.
JumpAbsolute	Repositions the record pointer to the specified record number.
JumpRelative	Repositions the record pointer x number of spaces from where it was last.
MoveFirst	Moves the record position to the first record in the recordset.
MoveLast	Moves the record position to the last record in the recordset.
MoveNext	Moves the record position to the record after the current record.
MovePrevious	Moves the record position to the record before the current record.
Refresh	Refreshes the current recordset.
Search	A non case-sensitive search on a particular column.
SetFilter	Applies a filter to the current recordset.

GetField

Retrieves the data for a particular column in a record.

Example

```
Sub GetField(column As Integer) As String
```

Parameters

- column - The column of the desired field

Return values

The data for a particular column in a record.

Remarks

JumpAbsolute

Repositions the record pointer to the specified record number.

Example

```
Sub JumpAbsolute(offset As Long)
```

Parameters

Return values

Remarks

JumpRelative

Repositions the record pointer x number of spaces from where it was last.

Example

```
Sub JumpAbsolute(offset As Long)
```

Parameters

Return values

Remarks

MoveFirst

Moves the record position to the first record in the recordset.

Example

```
Sub MoveFirst()
```

Parameters

Return values

Remarks

MoveLast

Moves the record position to the last record in the recordset.

Example

```
Sub MoveLast()
```

Parameters

Return values

Remarks

MoveNext

Moves the record position to the record after the current record.

Example

```
Sub MoveNext()
```

Parameters

Return values

Remarks

MovePrevious

Moves the record position to the record before the current record.

Example

```
Sub MovePrevious()
```

Parameters

Return values

Remarks

Refresh

Refreshes the current recordset.

Example

```
Sub Refresh()
```

Parameters

Return values

Remarks

Search

A non case-sensitive search on a particular column.

Example

```
Sub Search(column As Integer,searchStr As String)
```

Parameters

- column - Number indicating which column to search on
- searchStr - The string to search for

Return values

Remarks

SetFilter

Applies a filter to the current recordset.

Example

```
Sub SetFilter(filterType As Integer,parm1 As String,parm2 As
String)
```

Parameters

- filterType - ID of the filter

  1 = '|Filter by Severe Events

  2 = '|Filter by Warning Events

  3 = '|Filter by Informational Events

  4 = '|Filter by Diagnostic Events

  5 = '|Filter by All Events

- parm1 - Value of parameter 1 if the filter requires one or more parameters, otherwise pass in an empty string
- parm2 - Value of parameter 2 if the filter requires a second parameter, otherwise pass in an empty string

Return values

Remarks

Chapter 19

Preferences object

This object contains methods and properties used to get and set AppManager repository preferences.

This section covers the following topics:

- Preferences properties
- Preferences methods

Preferences properties

The following properties are available for the Preferences object.

Property	Description
AutoRefresh	A flag indicating whether the recordset will be requeried after any operation which could affect the recordset (useful for batch operations).
BOF	Indicates that the current record position is before the first record.
Count	Returns the number of records in a recordset.
EOF	Indicates the current record position is after the last record.
Error	Indicates if the last operation resulted in an error.
Fields	Returns an instance of a field's object.
LastError	Indicates the error code of the last operation.
SortIndex	Specifies which column in a recordset should be sorted on.
View	Sets/Gets the view ID which will be used in view-specific objects.

AutoRefresh

A flag indicating whether the recordset will be requeried after any operation which could affect the recordset (useful for batch operations).

Example

```
Property AutoRefresh() As BOOL
```

Parameters

Return values

A boolean expression that determines whether to automatically requery the recordset.

Remarks

BOF

Indicates that the current record position is before the first record.

Example

```
Property BOF() As BOOL
```

Parameters

Return values

A boolean expression.

Remarks

Read-only attribute.

Count

Returns the number of records in a recordset.

Example

```
Property Count() As Long
```

Parameters

Return values

The number of records in a recordset.

Remarks

Read-only attribute.

EOF

Indicates the current record position is after the last record.

Example

```
Property EOF() As BOOL
```

Parameters

Return values

A boolean expression.

Remarks

Read-only attribute.

Error

Indicates if the last operation resulted in an error.

Example

```
Property Error() As BOOL
```

Parameters

Return values

A boolean expression to indicate if an error occurred.

Remarks

Read-only attribute.

Fields

Returns an instance of a field's object.

Example

```
Property Fields() As Object
```

Parameters

Return values

An instance of a field's object.

Remarks

Read-only attribute.

LastError

Indicates the error code of the last operation.

Example

```
Property LastError() As Integer
```

Parameters

Return values

The error number of the last operation.

Remarks

Read-only attribute.

SortIndex

Specifies which column in a recordset should be sorted on.

Example

```
Property SortIndex() As Integer
```

Parameters

Return values

The current sort index number.

Remarks

View

Sets/Gets the view ID that will be used in view-specific objects.

Example

```
Property View() As Integer
```

Parameters

Return values

The number ID of the view to limit by.

Remarks

Preferences methods

The following methods are available for the Preferences object.

Method	Description
GetField	Retrieves the data for a particular column in a record.
GetReportTimeZoneBias	Returns the bias in seconds.
GetReportTimeZoneMode	Returns which time zone mode is used when generating reports.
JumpAbsolute	Repositions the record pointer to the specified record number.
JumpRelative	Repositions the record pointer x number of spaces from where it was last.
MoveFirst	Moves the record position to the first record in the recordset.
MoveLast	Moves the record position to the last record in the recordset.
MoveNext	Moves the record position to the record after the current record.
MovePrevious	Moves the record position to the record before the current record.
Refresh	Refreshes the current recordset.
Search	A non case-sensitive search on a particular column.
SetReportTimeZoneMode	Sets which time zone mode is used when generating reports.
SeverityToIcon	Indicates the color of a severity level icon.

GetField

Retrieves the data for a particular column in a record.

Example

```
Sub GetField(column As Integer) As String
```

Parameters

- column - The column of the desired field

Return values

The data for a particular column in a record.

Remarks

GetReportTimeZoneBias

Returns the bias in seconds.

Example

```
Sub GetReportTimeZoneBias() As Integer
```

Parameters

Return values

The bias in seconds.

Remarks

GetReportTimeZoneMode

Returns which time zone mode is used when generating reports.

Example

```
Sub GetReportTimeZoneMode() As Integer
```

Parameters

Return values

- 0 = Repository time zone
- 1 = Agent time zone
- 2 = Custom time zone

Remarks

JumpAbsolute

Repositions the record pointer to the specified record number.

Example

```
Sub JumpAbsolute(offset As Long)
```

Parameters

Return values

Remarks

JumpRelative

Repositions the record pointer x number of spaces from where it was last.

Example

```
Sub JumpAbsolute(offset As Long)
```

Parameters

Return values

Remarks

MoveFirst

Moves the record position to the first record in the recordset.

Example

```
Sub MoveFirst()
```

Parameters

Return values

Remarks

MoveLast

Moves the record position to the last record in the recordset.

Example

```
Sub MoveLast()
```

Parameters

Return values

Remarks

MoveNext

Moves the record position to the record after the current record.

Example

```
Sub MoveNext()
```

Parameters

Return values

Remarks

MovePrevious

Moves the record position to the record before the current record.

Example

```
Sub MovePrevious()
```

Parameters

Return values

Remarks

Refresh

Refreshes the current recordset.

Example

```
Sub Refresh()
```

Parameters

Return values

Remarks

Search

A non case-sensitive search on a particular column.

Example

```
Sub Search(column As Integer,searchStr As String)
```

Parameters

- column - Number indicating which column to search on
- searchStr - The string to search for

Return values

Remarks

SetReportTimeZoneMode

Sets which time zone mode is used when generating reports.

Example

```
Sub SetReportTimeZoneMode(TimeZoneMode As
Integer,TimeZoneBias As Integer)
```

Sample code

```
TimeZoneMode    = 2 ‘custom mode
TimeZoneBiasInMinutes = 180 ‘minutes
If timeZoneMode = 2 Then
   If TimeZoneBiasInMinutes > 0 Then
   nTimeZoneBias = nTimeZoneBiasInMinutes * 60
   End If
End If

PreferencesObject.SetReportTimeZoneMode timeZoneMode,
nTimeZoneBias
```

Parameters

- TimeZoneMode

  0 = Repository time zone

  1 = Agent time zone

  2 = Custom time zone

- TimeZoneBias

  The bias in seconds, used only in custome time zone mode (default = 0)

- Greenwich Mean Time

Return values

Remarks

SeverityToIcon

Indicates the color of a severity level icon.

Example

```
Sub SeverityToIcon(severity As Long) As Long
```

Parameters

- Severity - The severity value

Return values

- 1 = Severe icon (red)
- 2 = Warnig icon (yellow)
- 3 = Informational icon (blue)
- 4 = Diagnostic icon (magenta)

Remarks

Chapter 20

SrvGroups object

The SrvGroups object enables you to manipulate a Server Group and the servers within a group, as well as handle monitoring policies for a Server Group.

This section covers the following topics:

- SRVGroups properties
- SrvGroups methods

SRVGroups properties

The following properties are available for the SrvGroups object.

Property	Description
KSGroupName	Returns the member KS Group name at a specific index.
NewServerType	Get/Set the machine's type when adding a new server.
NumberOfKSGroups	Returns the number of member KS Groups in the Server Group.

KSGroupName

Returns the member KS Group name at a specific index.

Example

```
Property KSGroupName(nIndex) As String
```

Sample code

```
for x = 0 to SrvgroupObject.NumberOfKSGroups -1
   wscript.echo SrvgroupObject.KSGroupName (x)
next
```

Parameters

- nIndex - The index to a specific KS Group

Return values

The KS Group name.

Remarks

NewServerType

Get/Set the machine's type when adding a new server. The default is NT.

Example

```
Property NewServerType() As Integer
```

Parameters

Return values

- 0 = NT Server
- 1 = UNIX server

Remarks

NumberOfKSGroups

Returns the number of member KS Groups in the Server Group.

Example

```
Property NumberOfKSGroups As Integer
```

Parameters

Return values

The number of KS Groups applied to this Server Group.

Remarks

See LoadMemberKSG.vbx

SrvGroups methods

The following methods are available for the SrvGroups object.

Method	Description
AddToMonPolicy	Attaches a KS Group to a Monitoring Policy.
CreateServer	Creates a new server and returns the object ID.
CreateServerGroup	Creates a new server group and returns the object ID.
DeleteObject	Deletes an object by object ID and view ID.
DeleteServer	Deletes a server by object ID.
DeleteServerGroup	Deletes a server group by object ID.
LoadMonPolicy	Enumerates the member KS Groups of the Monitoring Policy on the current view.
MoveServer	Moves a server from one server group to another.
RemoveFromMonPolicy	Removes a KS Group from a Monitoring Policy.
SetFilter	Apply filter to the current recordset.
UpdateRestartCount	Changes the current restart count for the Monitoring Policy.

AddToMonPolicy

Attaches a KS Group to a Monitoring Policy.

Example

```
Sub AddToMonPolicy(srvGroupID as Long, viewID as Long,
KSGroupName as String, restartCount as Long)
```

Parameters

- srvGroupID - The Server Group's object ID
- viewID - The current view's ID
- KSGroupName - The name of the KS Group to add

- restartCount - The number of times to restart the jobs (applies only when there is no existing KS Group attached to the Monitoring Policy)

Return values

Remarks

`LoadMonPolicy()` must be invoked prior to calling this method.

CreateServer

Creates a new server and returns the object ID.

Example

```
Sub CreateServer(strServerName As null terminated wide
string,parentObjID As Long,checkFlags As 4 byte unsigned
int,bNTAutoDiscovery As Long) As Long
```

Parameters

- strServerName - Name of the new server
- parentObjID - object ID of the Server Group to which the new server belongs
- checkFlags
- bNTAutoDiscovery - flag to autorun Discovery_NT KS

Return values

The object ID of the new server; -1 if the operation fails.

Remarks

CreateServerGroup

Creates a new server group and returns the object ID.

Example

```
Sub CreateServerGroup(strGroupName As null terminated wide
string,parentObjID As Long,viewID As Long) As Long
```

Sample code

```
groupName = “SanJoseServers”
parentObjID = 1
viewID = 0
groupObjID = SrvGroupObject.CreateServerGroup(groupName,
parentObjID, viewID)
If SrvGroupObject.Error then
   WScript.Echo “ERROR: Could not add group”
Else
   WScript.Echo “Success: Group added successfully - object
   ID = “ & groupObjID
End If
```

Parameters

- strGroupName - Name of the new Server Group
- parentObjID - object ID of the Server Group to which the new Server Group belongs
- viewID - View ID of the new Server Group

Return values

The object ID of the new Server Group; -1 if the operation fails.

Remarks

DeleteObject

Deletes an object by object ID and view ID.

Example

```
Sub DeleteObject(objectID As Long,viewID As Long)
```

Parameters

- objectID - Object ID of the object to delete
- viewID - View ID of the object

Return values

Remarks

DeleteServer

Deletes a server by object ID.

Example

```
Sub DeleteServer(machObjID As Long,viewID As Long)
```

Parameters

- machObjID - Object ID of the server to delete
- viewID - View ID of the server to delete

Return values

Remarks

DeleteServerGroup

Deletes a server group by object ID.

Example

```
Sub DeleteServerGroup(srvGroupID As Long,viewID As Long)
```

Sample code

```
groupObjID = 1000
viewID = 0
SrvGroupObject.DeleteServerGroup groupObjID, viewID
If SrvGroupObject.Error then
   WScript.Echo “Error: Could not delete group”
Else
   WScript.Echo “Success: Group deleted successfully”
End If
```

Parameters

- srvGroupID - Object ID of the Server Group

- viewID - View ID of the Server Group

Return values

Remarks

See DeleteSrvGroup.vbs

LoadMonPolicy

Enumerates the member KS Groups of the Monitoring Policy on the current view.

Example

```
Sub LoadMonPolicy() As Integer
```

Parameters

Return values

The number of KS Groups.

Remarks

MoveServer

Moves a server from one Server Group to another.

Example

```
Sub MoveServer(objid As Long,oldParentObjID As
Long,newParentObjID As Long,viewID As Long) As Long
```

Parameters

- objid - Object ID of the server to move
- oldParentObjID - Object ID of the Server Group to which the server currently belongs
- newParentObjID - Object ID of the Server Group to which the server is being moved

- viewID - View ID of the Server Group

Return values

- 0 if successful
- -1 if failed

Remarks

RemoveFromMonPolicy

Removes a KS Group from a Monitoring Policy.

Example

```
Sub RemoveFromMonPolicy(srvGroupID as Long, viewID as Long,
KSGroupName as String)
```

Parameters

- srvGroupID - The Server Group's object ID
- viewID - The current view's ID
- KSGroupName - The name of the KS Group to remove

Return values

Remarks

`LoadMonPolicy()` must be invoked prior to calling this method.

SetFilter

Apply filter to the current recordset.

Example

```
Sub SetFilter(filterType As Integer,parm1 As String,parm2 As
String)
```

Parameters

- filterType - ID of the filter

1 = ‘ No filter

2 = ‘ view | filter by view

3 = ‘ parentobjid | filter by parent ID

- parm1 - Value of parameter 1 if the filter requires one or more parameters, otherwise pass in an empty string
- parm2 - Value of parameter 2 if the filter requires a second parameter, otherwise pass in an empty string

Return values

Remarks

UpdateRestartCount

Changes the current restart count for a Monitoring Policy.

Example

```
Sub UpdateRestartCount(srvGroupID as Long, viewID as Long,
newCount as Long)
```

Parameters

- srvGroupID - The Server Group’s object ID
- viewID - The current view’s ID
- newCount - The number of times to restart the jobs

Return values

Remarks

`LoadMonPolicy()` must be invoked prior to calling this method.

Chapter 21

Views object

This object represents the list of all discovered views.

This section covers the following topics:

- Views properties
- Views methods

Views properties

The following properties are available for the Views object.

Property	Description
AutoRefresh	A flag indicating whether the recordset will be requeried after any operation which could affect the recordset (useful for batch operations).
BOF	Indicates that the current record position is before the first record.
Count	Returns the number of records in a recordset.
EOF	Indicates the current record position is after the last record.
Error	Indicates if the last operation resulted in an error.
Fields	Returns an instance of a field's object.
KSGroupName	Returns the member KS Group name at a specified index.
LastError	Indicates the error code of the last operation.
NumberOfKSGroups	Returns the number of member KS Groups in the Server Group.
SortIndex	Specifies which column in a recordset should be sorted on.
ViewIconID	Returns the ID of the icon that is representative for a particular view.

AutoRefresh

A flag indicating whether the recordset will be requeried after any operation which could affect the recordset (useful for batch operations).

Example

```
Property AutoRefresh() As BOOL
```

Parameters

Return values

A boolean expression that determines whether to automatically requery the recordset.

Remarks

BOF

Indicates that the current record position is before the first record.

Example

```
Property BOF() As BOOL
```

Parameters

Return values

A boolean expression.

Remarks

Read-only attribute.

Count

Returns the number of records in a recordset.

Example

```
Property Count() As Long
```

Parameters

Return values

The number of records in a recordset.

Remarks

Read-only attribute.

EOF

Indicates the current record position is after the last record.

Example

```
Property EOF() As BOOL
```

Parameters

Return values

A boolean expression.

Remarks

Read-only attribute.

Error

Indicates if the last operation resulted in an error.

Example

```
Property Error() As BOOL
```

Parameters

Return values

A boolean expression to indicate if an error occurred.

Remarks

Read-only attribute.

Fields

Returns an instance of a field's object.

Example

```
Property Fields() As Object
```

Parameters

Return values

An instance of a field's object.

Remarks

Read-only attribute.

KSGroupName

Returns the member KS Group name at a specified index.

Example

```
Property KSGroupName(nIndex) As String
```

Sample code

```
for x = 0 to ViewsObject.NumberOfKSGroups -1
   wscript.echo SrvGroupObject.KSGroupName (x)
next
```

Parameters

- nIndex - The index to a specific KS Group

Return values

The KS Group name.

Remarks

LastError

Indicates the error code of the last operation.

Example

```
Property LastError() As Integer
```

Parameters

Return values

The error number of the last operation.

Remarks

Read-only attribute.

NumberOfKSGroups

Returns the number of member KS Groups in the Server Group.

Example

```
Property NumberOfKSGroups As Integer
```

Parameters

Return values

The number of KS Groups applied to the Server Group.

Remarks

SortIndex

Specifies which column in a recordset should be sorted on.

Example

```
Property SortIndex() As Integer
```

Parameters

Return values

The current sort index number.

Remarks

ViewIconID

Returns the ID of the icon that is representative for a particular view.

Example

```
Property ViewIconID() As Integer
```

Parameters

Return values

The icon ID.

Remarks

Views methods

The following methods are available for the Views object.

Method	Description
AddToMonPolicy	Attaches a KS Group to a Monitoring Policy.
GetField	Retrieves the data for a particular column in a record.
JumpAbsolute	Repositions the record pointer to the specified record number.
JumpRelative	Repositions the record pointer x number of spaces from where it was last.
LoadMonPolicy	Enumerates the member KS Groups of the Monitoring Policy on the current view.
MoveFirst	Moves the record position to the first record in the recordset.
MoveLast	Moves the record position to the last record in the recordset.
MoveNext	Moves the record position to the record after the current record.
MovePrevious	Moves the record position to the record before the current record.
Refresh	Refreshes the current recordset.
RemoveFromMonPolicy	Removes a KS Group from a Monitoring Policy.
Search	Revokes permission for a right on the backend.
SetFilter	Applies a filter to the current recordset.
UpdateRestartCount	Changes the current restart count for the Monitoring Policy.

AddToMonPolicy

Attaches a KS Group to a Monitoring Policy.

Example

```
Sub AddToMonPolicy(srvGroupID as Long, viewID as Long,
KSGroupName as String, restartCount as Long)
```

Parameters

- srvGroupID - The Server Group's object ID
- viewID - The current view's ID
- KSGroupName - The name of the KS Group to add
- restartCount - The number of times to restart the jobs (applies only when there is no existing KS Group attached to the Monitoring Policy)

Return values

Remarks

`LoadMonPolicy()` must be invoked prior to calling this method.

GetField

Retrieves the data for a particular column in a record.

Example

```
Sub GetField(column As Integer) As String
```

Parameters

- column - The column of the desired field

Return values

The data for a particular column in a record.

Remarks

JumpAbsolute

Repositions the record pointer to the specified record number.

Example

```
Sub JumpAbsolute(offset As Long)
```

Parameters

Return values

Remarks

JumpRelative

Repositions the record pointer x number of spaces from where it was last.

Example

```
Sub JumpAbsolute(offset As Long)
```

Parameters

Return values

Remarks

LoadMonPolicy

Enumerates the member KS Groups of the Monitoring Policy on the current view.

Example

```
Sub LoadMonPolicy() As Integer
```

Parameters

Return values

The number of KS Groups.

Remarks

MoveFirst

Moves the record position to the first record in the recordset.

Example

```
Sub MoveFirst()
```

Parameters

Return values

Remarks

MoveLast

Moves the record position to the last record in the recordset.

Example

```
Sub MoveLast()
```

Parameters

Return values

Remarks

MoveNext

Moves the record position to the record after the current record.

Example

```
Sub MoveNext()
```

Parameters

Return values

Remarks

MovePrevious

Moves the record position to the record before the current record.

Example

```
Sub MovePrevious()
```

Parameters

Return values

Remarks

Refresh

Refreshes the current recordset.

Example

```
Sub Refresh()
```

Parameters

Return values

Remarks

RemoveFromMonPolicy

Removes a KS Group from a Monitoring Policy.

Example

```
Sub RemoveFromMonPolicy(srvGroupID as Long, viewID as Long,
```

```
KSGroupName as String)
```

Parameters

- srvGroupID - The Server Group's object ID
- viewID - The current view's ID
- KSGroupName - The name of the KS Group to remove

Return values

Remarks

`LoadMonPolicy()` must be invoked prior to calling this method.

Search

Revokes permission for a right on the backend.

Example

```
Sub RevokeRight(strUser As String,rightID As Long)
```

Parameters

- column - Number indicating which column to search on
- searchStr - The string to search for

Return values

Remarks

SetFilter

Applies a filter to the current recordset.

Example

```
Sub SetFilter(filterType As Integer,parm1 As String,parm2 As
String)
```

Parameters

- filterType - ID of the filter

  1 = ‘ No Filter

  2 = ‘ view name | filter by view name

- parm1 - Value of parameter 1 if the filter requires one or more parameters, otherwise pass in an empty string
- parm2 - Value of parameter 2 if the filter requires a second parameter, otherwise pass in an empty string

Return values

Remarks

UpdateRestartCount

Changes the current restart count for a Monitoring Policy.

Example

```
Sub UpdateRestartCount(srvGroupID as Long, viewID as Long,
newCount as Long)
```

Parameters

- srvGroupID - The Server Group’s object ID
- viewID - The current view’s ID
- newCount - The number of times to restart the jobs

Return values

Remarks

`LoadMonPolicy()` must be invoked prior to calling this method.

Index