24x7 Remote Control COM API Reference

24x7 Scheduler™ Multi-platform Edition 5.1

Table of contents

About This Manual

This user manual describes all features of the 24x7 Remote Control COM API for 24x7 Scheduler™ Windows Edition and 24x7 Scheduler™ Multi-platform Edition. This manual contains information for developers and experienced users of the 24x7 Scheduler who want to create custom interfaces to the 24x7 Scheduler system and/or tightly integrate it with other applications and systems.

The complete 24x7 Scheduler documentation consists of several manuals and also an on-line interactive help system. The on-line help is available at any time when you are running 24x7 Scheduler. Depending on what you are doing, you can press F1, select Help from the Menu Bar, or select the Help button on a dialog.

Conventions Used in This Document

This section describes the style conventions used in this document.

Italic
An *italic* font is used for filenames, URLs, emphasized text, and the first usage of technical terms.

`Monospace`
A `monospaced` font is used for code fragments and data elements.

Bold
A **bold** font is used for important messages, names of options, names of controls and menu items, and keys.

Graphical marks

 **-** This mark is used to indicate product specific options and issues and to mark useful tips.

 **-** This mark is used to indicate important notes.

Abbreviations and Terms

This guide uses common abbreviations for many widely used technical terms including COM, ASP, API, and other.

Trademarks

24x7 Automation Suite, 24x7 Scheduler, DB Audit, DB Audit Expert, DB Mail, DB Tools for Oracle are trademarks of SoftTree Technologies, Inc.

Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP are registered trademarks of Microsoft Corporation. UNIX is registered trademark of the X/Open Consortium. Sun, SunOS, Solaris, SPARC are trademarks or registered trademarks of Sun Microsystems, Inc. Ultrix, Digital UNIX and DEC are trademarks of Digital Equipment Corporation. HP-UX is a trademark of Hewlett-Packard Co. IRIX is a trademark of Silicon

Overview

About the Component Object Model

The Microsoft Component Object Model (COM) defines a standard way for software components to supply services to each other. 24x7 Scheduler includes a build-in COM automation server called **24x7 Remote Control COM** that provides access to the most features of the 24x7 Scheduler. Applications created with COM-compliant tools such as Visual Basic, Delphi, PowerBuilder, Java and other, can make use of the business logic in the 24x7 COM object by creating an instance of the object and calling the methods exposed in its interface. 24x7 Remote Control COM uses a dispatch interface (also called *dispinterface*) that allows COM-compliant tools to invoke methods on the 24x7 COM server using the Invoke method of a standard COM interface called IDispatch.

Using 24x7 COM API in Your Program

The following steps summarize typical programming tasks required to incorporate the functionality of the 24x7 Scheduler API in a client application if you are using the COM implementation.

To code a client application

1. Create an instance of a COM object by calling the **CreateObject** function (or similar function available in your programming language). Then connect to an active 24x7 Scheduler server or 24x7 Remote Agent by calling the **OpenSession** method of the COM object. For more information about connecting to 24x7 remote servers, see OpenSession method description. For information on available COM interfaces and objects see Supported COM Interfaces topic.
2. After creating an instance of a COM object for a specific server call necessary API methods for performing the required tasks. For more Information see Object Methods topic.
3. Terminate the connection to the 24x7 Scheduler server by calling the **CloseSession** method of the COM object. For more information, see CloseSession topic.

For more information about managing 24x7 Schedulers and scheduled jobs see the 24x7 Scheduler User's Guide.

A Quick Example: Visual Basic application

The following paragraph provides a quick example of calling **24x7 Remote Control COM** from Visual Basic code.

1. Declare 24x7 COM object variable

```
Dim Remote_24x7 As Object
Dim RC As Integer
```

2. Create 24x7 Remote Control, using its programmatic identifier (ProgID="24x7 Remote Control") and check that the creation was successful:

```
Set Remote_24x7 = CreateObject("24x7 Remote Control")

If Remote_24x7 Is Nothing Then
  ' Handle the error ...
  MsgBox "Error creating 24x7 COM object..."
End If
```

3. Access functions or properties of the 24x7 Remote Control using automation syntax:

```
' Open new session
RC = Remote_24x7.OpenSession("john_doe", "password", _
                    "WinSock", "LocalHost", "1096", "", False)
If RC <> 1 Then
    ' Handle the error
    MsgBox Remote_24x7.LastError
Else
    ' Get job list
    Dim buffer As String
    RC = Remote_24x7.GetJobList(buffer, True)
    ' Change start time for job #12 to 6:00 AM
    RC = Remote_24x7.SetJobProperty("12", "START_TIME", _
                                          "6:00")
    ' Done. Close session
    RC = Remote_24x7.CloseSession()
End If

' Destroy COM object and release all allocated resources
Set Remote_24x7 = Nothing
```

24x7 Remote Control COM API

Supported COM Interfaces

24x7 Scheduler provides 3 different COM interfaces which allow using 24x7 API with virtually any application supporting Component Object Model. Second and third interfaces are actually implemented as wrappers around the first standard interface. They are designed for use in environments that provide only limited support for Component Object Model.

Below are descriptions of the supported interfaces.

1. COM interface for use with Visual Basic, C/C++, Delphi, PowerBuilder and other programming environments. In the following topics we will refer to this interface as **standard COM** interface.

   To instantiate 24x7 COM interface create "24x7 Remote Control" object.
   Visual Basic example:

   ```
   ' Declare COM object variable
   Dim Remote_24x7 As Object

   ' Create 24x7 Remote Control, using its programmatic
   ' identifier (ProgID="24x7 Remote Control")
   Set Remote_24x7 = CreateObject("24x7 Remote Control")
   ```

2. COM Interface for use with VBScript and other compatible programming environments. In the following topics we will refer to this interface as **VBScript COM** interface.

   To instantiate 24x7 COM interface create "24x7 Remote Control" object.
   Visual Basic example:

   ```
   ' Declare COM object variable
   Dim Remote_24x7

   ' Create 24x7 Remote Control, using its programmatic
   ' identifier (ProgID="w24x7ASP.vbRemote24x7")
   Set Remote_24x7 = CreateObject("w24x7ASP.vbRemote24x7")
   ```

3. COM Interface for use with JavaScript and other compatible programming environments. In the following topics we will refer to this interface as **JavaScript COM** interface.

   To instantiate 24x7 COM interface create "24x7 Remote Control" object.
   Visual Basic example:

   ```
   ' Declare COM object variable
   Dim Remote_24x7

   ' Create 24x7 Remote Control, using its programmatic
   ' identifier (ProgID="w24x7ASP.jsRemote24x7")
   Set Remote_24x7 = CreateObject("w24x7ASP.jsRemote24x7")
   ```

All 24x7 COM interfaces provide the same set of properties and methods. However declarations and return values of some methods differ in different interfaces. These differences are described in detail in the following topics.

Which Interface Is Right For Me?

If your programming environment provides full support for Component Object Model including support for passing and returning parameters by references for such basic data-types as String and Long use the standard 24x7 COM interface whose programmatic identifier is **24x7 Remote Control**.

Most full-featuring programming systems such as Visual Basic, Visual Basic.NET, C/C++, C#, Delphi, PowerBuilder and many other provide such support and therefore application developed with those systems can use the standard interface.

If your programming system supports only Variant data-types and also supports passing and returning Variant data-type parameters by references use VBScript wrapper for 24x7 COM interface whose full programmatic identifier is **w24x7ASP.vbRemote24x7**.

This interface can be used with any program using VBScript or compatible programming language, including VBScript in server side and client side ASP pages.

If your programming system does not support passing and returning parameters by references use JavaScript wrapper for 24x7 COM interface whose full programmatic identifier is **w24x7ASP.jsRemote24x7**.

This interface can be used with any program using JavaScript or compatible programming language, including JavaScript in server side and client side ASP pages.

Object Constants

Long RC_ERROR Long RC_WARNING Long RC_INFO	These constants can be used with the LogMessage function for the message Severity parameter. Use RC_ERROR to write an error message. Use RC_WARNING to write an warning message. Use RC_INFO to write all other informational messages.
Long SECURITY_OFF Long ROLE_ADMIN Long ROLE_STANDARD Long ROLE_RESTRICTED Long ROLE_GUEST	These constants can be used with the GetUserRole function for the return value. For more details see description of the GetUserRole function.

Object Properties

All properties are read-only. Do not attempt to modify their values directly.

Boolean **ConnectedToRemote**	Indicates session status TRUE – session is opened FALSE – session is closed See OpenSession function for more details.
String **LastError**	Stores the error text for the last detected error. You can use this property to obtain the error text when a function's return value indicates that it failed. Retrieve **LastError** immediately after a function call fails.
String **Terminal**	Network name of the computer where 24x7 COM+ interface is invoked.
String **UserID**	User name passed as a parameter to the last called OpenSession function. If an empty string has been passed, **UserID** contains network name of the user logged on to the computer where 24x7 COM+ interface is invoked.
String **ipStringBuffer**	This property is available in the JavaScript COM interface. It is used to pass character type return values that are normally passed by reference in standard and VBScript

	COM interfaces.
Long **ipLongBuffer**	 This property is available in the JavaScript COM interface. It is used to pass numeric type return values that are normally passed by reference in standard and VBScript COM interfaces.

Object Methods

All supported COM interface methods are described in the following topics. For your convenience methods are listed in alphabetical order. Special notices are used in places where method parameters or return values differ in different 24x7 COM interface versions. For description of available COM interfaces see Supported COM Interfaces topic.

AddAgentProfile

Long AddAgentProfile (String ProfileName,
String ComMethod,
String Location,
String Port,
String Options)

The **AddAgentProfile** function creates new 24x7 Remote Agent or 24x7 Master Scheduler profile.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

ProfileName	Name of the new Remote Agent Profile. This name must be unique.
ComMethod	Name of the communication driver to be used for the connections to the Agent Values are: • WinSock • NamedPipes
Location	The location of the Agent. For the **WinSock** driver specify one of the following values: • The IP address (for example, 199.99.99.91) • The host name of the remote computer (network computer name in workgroup) • LocalHost (this indicates that the Agent resides on the local machine) For the **NamedPipes** driver specifies the location portion of the pipe name. The combination of the Location and Application values forms the pipe name. The pipe name is constructed as follows: \\location\PIPE\application. If no location is specified, a local pipe name is constructed using a dot (.) in the machine name portion.
Port	For the **WinSock** driver specify either of the following: • The port number for the Agent (for example, 1096). Each server application requires a unique port number on the server machine. If you specify a port number, select a number that is greater than 4096 and less than 65536. • The service name for Agent. The service name is an indirect reference to the port number. The mapping of the service name to the port number is specified in the TCP/IP services file. Normally, this file (SERVICES) is located in the Windows directory for Win95/98/Me and in the C:\WINNT\SYSTEM32\DRIVERS\ETC directory for

	Windows NT4/2000/XP/2003/Vista/2008. You may need to edit this file manually to add 24x7 Port or Service. For the **NamedPipes** driver specify the application portion of the pipe name. The combination of the Location and Application values forms the pipe name. The pipe name is constructed as follows: *\\location\PIPE\application*. The Application must be unique for the Location you specify.
Options	Specifies one or more additional communications options. If you specify more than one option, you need to separate the options with commas. For the Local driver this property is ignored. **BufSize=n** Sets the connection buffer size to the value specified. **MaxListeningThreads=n** Determines the maximum number of listening threads available on the Master Scheduler and Remote Agents. **MaxRetry=n** Specifies how many times the Remote Control will try to connect when the Agent's listening port is busy. Applies to the **WinSock** driver only. **NoDelay=1** Specifies that each packet be sent without delay. Corresponds to the TCP_NODELAY option. Setting this option may degrade performance significantly. Do not use this option unless you thoroughly understand its implications. Applies to the **WinSock** driver only. **RawData=1** Specifies that raw data be passed over the network. By default, the **WinSock** driver obscures the data that is passed over the network. Setting this option to 1 overrides the default behavior. Both the Remote Control and Agent must have the same setting. If there is a discrepancy between the Remote Control and Agent's settings, the communication fails. Setting this option to 1 may improve performance slightly. Applies to the **WinSock** driver only.

See also:

UpdateAgentProfile
DeleteAgentProfile
GetAgentList

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.AddAgentProfile( "Print Server", "Winsock", _
                               "192.168.100.1", "1096", "")
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
```

```
Dim RetCode
RetCode = obj.AddAgentProfile( "Print Server", "Winsock", _
                               "192.168.100.1", "1096", "")
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = 0;
RetCode = obj.AddAgentProfile( "Print Server", "Winsock",
                               "192.168.100.1", "1096", "");
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

AddDatabaseProfile

Long AddDatabaseProfile (String ProfileName,
String DatabaseDriver,
String Server,
String DatabaseName,
Boolean AutoCommit,
String User,
String Password)

The **AddDatabaseProfile** function creates new database profile.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

ProfileName	Name of the new Database Profile. This name should be unique.
DatabaseDriver	The name of the database driver that you want to use for connection. This name must much any supported DBMS name as they are specified in the Database Profiles dialog in 24x7 Scheduler (see Tools/Database Profiles menu).
Server	(Optional) The database server name. The format for the name differs for different database systems. Specify an empty string "" if you connect to a local database and the server name is not required for connection. If you connect using ODBC driver (DatabaseDriver="ODBC"), specify name of the desired ODBC profile as the Server name.
DatabaseName	(Optional) The name of the database. Specify an empty string "" if the database name is not required for connection.
AutoCommit	AutoCommit mode. Some databases support AutoCommit mode. Specify TRUE to turn AutoCommit on or specify FALSE otherwise. If DBMS does not support AutoCommit then this parameter is ignored.
User	The name of the user to log on to the database.
Password	The password to use to log on to the database.

See also:

UpdateDatabaseProfile
GetDatabaseProfile
DeleteDatabaseProfile
GetDatabaseList

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.AddDatabaseProfile( "Prod Server", _
             "MS SQL Server 7.x and later", "Neptune", _
             "dataware", True, "sa", "******")
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.AddDatabaseProfile( "Prod Server", _
             "MS SQL Server 7.x and later", "Neptune", _
             "dataware", True, "sa", "******")
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = 0;
RetCode = obj.AddDatabaseProfile( "Prod Server",
             "MS SQL Server 7.x and later", "Neptune",
             "dataware", True, "sa", "******");
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

AddHoliday

Long AddHoliday (Long Year,
Long Month,
Long Day,
String Description)

In 24x7 Scheduler Windows Edition, the **AddHoliday** function adds new holiday record to the 24x7 Scheduler Holiday Table.

In 24x7 Scheduler Multi-platform Edition this function adds new exception date to the **[default]** calendar.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

Year	Year part of the holiday date. Supported range: 1900-3000
Month	Month part of the holiday date. Supported range: 1-12
Day	Day part of the holiday date. Supported range: 1-31. Day must be a valid day for the specified Year and Month.
Description	Holiday name or description limited by 50 characters.

See also:

DeleteHoliday
GetHolidays

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.AddHoliday( 2003, 11, 27, "Thanksgiving Day")
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.AddHoliday( 2003, 11, 27, "Thanksgiving Day")
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
```

```
var RetCode = 0;
RetCode = obj.AddHoliday( 2003, 11, 27, "Thanksgiving Day");
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

AddJobQueue

Long AddJobQueue (String QueueName,
Long MaxSize)

The **AddJobQueue** function creates new job queue.

 **Note:** In 24x7 Scheduler Windows Edition, adding new job queue does not have an immediate effect. The new queue is available to jobs only after the 24x7 Scheduler is restarted. In 24x7 Scheduler Multi-platform Edition, the new queue is available immediately to run jobs.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

QueueName	Name of the new Job Queue. This name should be unique.
MaxSize	Max queue size in Mbytes. 1 Mbytes should be sufficient in most cases.

See also:

UpdateJobQueue
GetJobQueue
DeleteJobQueue
GetJobQueueList

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.AddJobQueue( "Payroll jobs", 1)
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.AddJobQueue( "Payroll jobs", 1)
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = 0;
RetCode = obj.AddJobQueue( "Payroll jobs", 1);
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

AddJobQueueEx

Long AddJobQueueEx (String QueueName,
Long MaxSize,
Long MaxJobs,
Boolean NearCapacityAlerts,
String EmailSender,
String EmailPassword,
String EmailRecipients)

The **AddJobQueueEx** function creates new job queue.

 **Note:** In 24x7 Scheduler Windows Edition, adding new job queue does not have an immediate effect. The new queue is available to jobs only after the 24x7 Scheduler is restarted. In 24x7 Scheduler Multi-platform Edition, the new queue is available immediately to run jobs.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

QueueName	Name of the new Job Queue. This name should be unique.
MaxSize	Maximum allowed disk space usage in Mbytes. 1 Mbytes should be sufficient in most cases. As of 24x7 Scheduler Multi-platform Edition v4.3, this queue parameter is reserved for future use. The parameter is used in 24x7 Scheduler Windows Edition only. Specify zero value for unlimited disk space usage.
MaxJobs	Maximum number of jobs allowed in a queue at any moment in time. The count includes both running and queued jobs, including jobs placed on hold. Specify zero value for unlimited number of jobs.
NearCapacityAlerts	Enables sending email alerts in case a queue is at or near its maximum capacity. The capacity is controlled by **MaxSize** and **MaxJobs** parameters
EmailSender	Email account (for MAPI email protocol) or email address (for SMTP email protocol) to use for sending "near capacity" and "over capacity" email alerts.
EmailPassword	Email password to use for authenticating email sender to the email server. Specify an empty string or null value if password is not required.
EmailRecipients	Comma-separated list of email recipient addresses for "near capacity" and "over capacity" email alerts.

See also:

UpdateJobQueueEx
GetJobQueue
DeleteJobQueue
GetJobQueueList

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.AddJobQueueEx( "Payroll jobs", 1, 100,
              "alerter@domain.com", "", "helpdesk@domain.com" )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.AddJobQueueEx( "Payroll jobs", 1, 100,
              "alerter@domain.com", "", "helpdesk@domain.com")
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = 0;
RetCode = obj.AddJobQueueEx( "Payroll jobs", 1, 100,
              "alerter@domain.com", "", "helpdesk@domain.com");
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

AddTemplate

Long AddTemplate (String Section,
String Name,
String Template,
String Text)

The **AddTemplate** function creates new job template.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

Section	Name of the template section in the catalog. **AddTemplate** automatically creates the specified **Section** if it does not yet exist.
Name	Name of the new template. This name must be unique within the **Section**.
Template	Name of the new template file. The file name must be unique.
Text	Template text.

See also:

DeleteTemplate
SetTemplate
GetTemplate
GetJobTemplateData
GetTemplateCatalog
SetJobTemplateData

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
Dim TemplateCode AS String

' Copy template code from the text editor control
TemplateCode = TemplateForm.Editor.Text
' Create new template using template code
RetCode = obj.AddTemplate( "FTP jobs", "Upload Monthly Reports",
                           "%HOME%\Templates\ftp_month_reports.ini", _
                            TemplateCode )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
```

```
Dim RetCode
Dim TemplateCode

' Obtain template code from the submitted form
TemplateCode = Request.Form("TEMPLATE_CODE")
' Create new template using template code
RetCode = obj.AddTemplate( "FTP jobs", "Upload Monthly Reports",
                           "%HOME%\Templates\ftp_month_reports.ini", _
                            TemplateCode )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = 0;
// Obtain template code from the submitted form
var TemplateCode = Request.Form("TEMPLATE_CODE");
// Create new template using template code
RetCode = obj.AddTemplate( "FTP jobs", "Upload Monthly Reports",
                           "%HOME%\\Templates\\ftp_month_reports.ini",
                            TemplateCode );
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

ChangeFolder

Long ChangeFolder (String JobID,
String TargetFolderID)

The **ChangeFolder** function moves a job with the specified **JobID** to a job folder specified by the **TargetFolderID** value.
Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name.
TargetFolderID	Folder ID converted to string or folder name.

See also:

CreateJob
SetJobProperty
CreateFolder

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
' Move job #624 to folder with id -1
RetCode = obj.ChangeFolder( "624", "-7")
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
' Move job "My job" to folder "My folder"
RetCode = obj.ChangeFolder( "My job", "My folder")
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
' Move job #624 to folder with id -1
RetCode = obj.ChangeFolder( "624", "-7")
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
' Move job "My job" to folder "My folder"
RetCode = obj.ChangeFolder( "My job", "My folder")
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = 0;
// Move job #624 to folder with id -1
RetCode = obj.ChangeFolder( "624", "-7");
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
// Move job "My job" to folder "My folder"
```

```
RetCode = obj.ChangeFolder( "My job", "My folder");
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

CloseSession

Long CloseSession ( )

The **CloseSession** function closes work session with the 24x7 Master Scheduler or 24x7 Remote Agent and terminates the connection. The session must be previously opened using the **OpenSession** function.
The **CloseSession** function does not destroy **24x7 Remote Control COM** object. The object can be reused to open another session. You must destroy the object yourself using the appropriate command of the environment in which **24x7 Remote Control COM** was created. For example in Visual Basic it is **Set *object* = Nothing**. Failure to close the session may cause your program to hang or crash. Failure to destroy the COM object may lead to a memory leak.

 If you destroy COM object and a session is still open, the COM object automatically calls **CloseSession**. In web based applications object clean up and destruction is often left to the web server so that in such applications you usually do not need to call **CloseSession** explicitly.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters: None

See also:

OpenSession
Using 24x7 COM API in Your Program

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' create COM object
Dim obj As Object
Set obj = CreateObject("24x7 Remote Control")
' open session
obj.OpenSession( "john_doe", "password", "WinSock", _
             "LocalHost", "1096", "", False)
' ... do something here ...
' close session
obj.CloseSession()
' destroy COM object
Set obj = Nothing
```

2. ASP example using VBScript COM interface

```
' create COM object
Dim obj
Set obj = Server.CreateObject("w24x7ASP.vbRemote24x7")
' open session
obj.OpenSession( "john_doe", "password", "WinSock", _
             "LocalHost", "1096", "", False)
' ... do something here ...
' close session destroy COM object
```

```
Set obj = Nothing
```

3. ASP example using JavaScript COM interface

```
// create COM object
var obj = Server.CreateObject("w24x7ASP.jsRemote24x7");
// open session
obj.OpenSession( "john_doe", "password", "WinSock",
             "LocalHost", "1096", "", False);
// ... do something here ...
// close session and destroy COM object
delete obj
```

CreateFolder

Standard COM and VBScript COM interfaces:
Long CreateFolder (String FolderName,
String FolderDescription,
ByRef Long FolderID)

JavaScript COM:
Long CreateFolder (String FolderName, String FolderDescription)

The **CreateFolder** function creates new job folder.

Return: Returns 1 if the function succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

If the function succeeds, in standard and VBScript COM interfaces the **FolderID** parameter is populated with the ID of the new folder.
In JavaScript COM interface this parameter is not used. To obtain the ID of newly created folder use ipLongBuffer property.

Parameters:

FolderName	The name of the new folder
FolderDescription	The description that you want to attach to the folder so other people can see how this folder should be used. You can specify an empty string if no description is needed.
FolderID This parameter is not available in JavaScript COM interface	This parameter is passed by reference. If the function succeeds, it updates FolderID with the new unique number that can be used in other functions as a unique folder identifier.

See also:

DeleteFolder
GetFolderProperty
SetFolderProperty

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, NewFolderID As Integer
RetCode = obj.CreateFolder( "Weekly Payroll", _
      "Store payroll related jobs in this folder", NewFolderID)
If RetCode <> 1 Then
      ' Display error message
      MsgBox obj.LastError
```

```
Else
       MsgBox "Weekly Payroll folder id is " & CStr(NewFolderID)
End If
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, NewFolderID
RetCode = obj.CreateFolder( "Weekly Payroll", _
       "Store payroll related jobs in this folder", NewFolderID)
If RetCode <> 1 Then
       ' Display error message
       Response.Write(obj.LastError)
Else
       Response.Write("Weekly Payroll folder id is " & _
                      CStr(NewFolderID))
End If
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.CreateFolder( "Weekly Payroll", _
       "Store payroll related jobs in this folder");
if (RetCode != 1)
       /* Display error */ Response.Write(obj.LastError);
else
       Response.Write("Weekly Payroll folder id is " +
                      obj.ipLongParm);
```

CreateJob

Long CreateJob (String JobDefinition)

The **CreateJob** function creates new job and adds it to the active job pool.

Return: Job ID of the created job as a positive number or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

JobDefinition	The Job definition in JDL format. For a wide variety of examples see job templates available in the [24x7 install directory]\Template subdirectory. The default path is *C:\Program Files\24x7 Automation 3\Template*. For supported job properties and their JDL names see 24x7 User's Guide. As a quick reference see Job Properties in JDL Format topic in this manual.

See also:

SetJobProperty
SetJobTemplateData
ChangeFolder
UpdateJob
DisableJob
ProtectJob
RunJob

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, JobDefinition AS String

' Copy job definition from text editor control
JobDefinition = JobDefinition.Editor.Text
' Create new job
RetCode = obj.CreateJob( JobDefinition )
If RetCode < 0 Then
       ' Display error message
       MsgBox obj.LastError
Else
       MsgBox "Job ID: " & CStr(RetCode)
End If
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, JobDefinition

' Get job definition from the submitted form
```

```
JobDefinition = JobDefinition.Editor.Text
' Create new job
RetCode = obj.CreateJob( JobDefinition )
If RetCode < 0 Then
       ' Display error message
       Response.Write(obj.LastError)
Else
       Response.Write("Job ID: " & _
                      CStr(RetCode))
End If
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = 0;
// Get job definition from the submitted form
var JobDefinition = Request.Form("JOB_DEF");
// create new job
RetCode = obj.CreateJob( JobDefinition );
if (RetCode < 0)
       /* Display error */ Response.Write(obj.LastError);
else
       Response.Write("Job ID: " + RetCode);
```

DeleteAgentProfile

Long DeleteAgentProfile (String ProfileName)

The **DeleteAgentProfile** function deletes existing profile of 24x7 Remote Agent or 24x7 Master Scheduler.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

ProfileName	Name of an existing Remote Agent Profile.

See also:

AddAgentProfile
UpdateAgentProfile

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.DeleteAgentProfile( "QA Server" )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.DeleteAgentProfile( "QA Server" )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.DeleteAgentProfile( "QA Server" );
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

DeleteDatabaseProfile

Long DeleteDatabaseProfile (String ProfileName)

The **DeleteDatabaseProfile** function deletes existing database profile.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

ProfileName	Name of an existing Database Profile.

See also:

AddDatabaseProfile
UpdateDatabaseProfile

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.DeleteDatabaseProfile( "QA2" )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.DeleteDatabaseProfile( "QA2" )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.DeleteDatabaseProfile( "QA2" );
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

DeleteFolder

Long DeleteFolder (String FolderID)

The **DeleteFolder** function deletes job folder from 24x7 Job Database. If the specified folder contains jobs, all these jobs are also deleted.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

FolderID	Folder ID converted to string or folder name.

See also:

CreateFolder
SetFolderProperty

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.DeleteFolder( "Payroll jobs" )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.DeleteFolder( "Payroll jobs" )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.DeleteFolder( "Payroll jobs" );
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

DeleteHoliday

Long DeleteHoliday (Long Year,
Long Month,
Long Day)

In 24x7 Scheduler Windows Edition, the **DeleteHoliday** function deletes matching holiday record from the 24x7 Scheduler Holiday Table.

In 24x7 Scheduler Multi-platform Edition this function deletes matching date from the **[default]** calendar.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

Year	Year part of the holiday date. Supported range 1900-3000
Month	Month part of the holiday date. Supported range 1-12
Day	Day part of the holiday date. Supported range 1-31. Day must be a valid day for the specified Year and Month.

See also:

AddHoliday
GetHolidays

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.DeleteHoliday( 2003, 11, 25 )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.DeleteHoliday( 2003, 11, 25 )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.DeleteHoliday( 2003, 11, 25 );
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

DeleteJob

Long DeleteJob (String JobID)

The **DeleteJob** function deletes job from both the active job pool and the 24x7 Job Database.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name.

See also:

DisableJob
CreateJob
DeleteFolder

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.DeleteJob( "246" )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.DeleteJob( "246" )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.DeleteJob( "246" );
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

DeleteJobQueue

Long DeleteJobQueue (String QueueName)

The **DeleteJobQueue** function deletes existing job queue.

Note: Deleting job queue does not have an immediate effect. It takes effect only after the 24x7 Scheduler is restarted.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

QueueName	Name of an existing Job Queue.

See also:

AddJobQueue
UpdateJobQueue
GetJobQueue
GetJobQueueList

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.DeleteJobQueue( "Weekly processing" )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.DeleteJobQueue( "Weekly processing" )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = 0;
RetCode = obj.DeleteJobQueue( "Weekly processing" );
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

DeleteTemplate

Long DeleteTemplate (String Section,
String Name,
Boolean DeleteFile))

The **DeleteTemplate** function deletes template from the template catalog (TEMPLATE.INI) and optionally deletes the template file.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

Section	Name of the template section in the template catalog from which to delete the template specified by **Name** parameter.
Name	Name of the template to be deleted.
DeleteFile	If **DeleteFile** parameter is set to TRUE the template source file is also deleted. If **DeleteFile** is set to FALSE the template name is deleted from the catalog, but the file is left on disk.

See also:

AddTemplate
GetTemplate
GetTemplateCatalog
SetTemplate
SetJobTemplateData

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.DeleteTemplate( "Web reports", "FTP upload", False )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.DeleteTemplate( "Web reports", "FTP upload", False )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = 0;
```

```
RetCode = obj.DeleteTemplate( "Web reports", "FTP upload", false );
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

DisableJob

Long DisableJob (String JobID)

The **DisableJob** function disables job and removes it from both the active job pool. The job is not deleted from the 24x7 Job Database and can be later enabled again.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name

See also:

EnableJob
DeleteJob

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.DisableJob( "246" )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.DisableJob( "246" )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.DisableJob( "246" );
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

EnableJob

Long EnableJob (String JobID)

The **EnableJob** function enables job and places it backs to the active job pool.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name.

See also:

DisableJob
DeleteJob

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.EnableJob( "246" )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.EnableJob( "246" )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.EnableJob( "246" );
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

GetAgentList

Standard COM and VBScript COM interfaces:
Long GetAgentList (ByRef String Buffer,
Boolean HTMLFormat)

JavaScript COM:
Long GetAgentList (Boolean HTMLFormat)

The **GetAgentList** function obtains list of names of configured 24x7 Remote Agent Profiles.

If the function succeeds, in standard and VBScript COM interfaces the **Buffer** variable is populated with the name list. If **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as a HTML table; otherwise it is populated with a plain text containing each entry on a new line.

In JavaScript COM interface the **Buffer** parameter is not used. To obtain the list of agent names use ipStringBuffer property. If **HTMLFormat** is TRUE, the **ipStringBuffer** property is populated with a text formatted as a HTML table; otherwise it is populated with a plain text containing each entry on a new line.

Note: You can use GetToken function to parse the returned list and extract individual elements or convert it into an array of elements. For details see description and examples for **GetToken** function. In JavaScript you can also use the built-in *`split()`* function to convert the returned list into an array of elements.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

Buffer This parameter is not available in JavaScript COM interface	A String variable passed by reference. In case of successful function execution, the profile list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the list of 24x7 Remote Agent Profile names. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:

AddAgentProfile
DeleteAgentProfile
GetAgentProfile

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, NameList As String
RetCode = obj.GetAgentList( NameList, False )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, NameList
RetCode = obj.GetAgentList( NameList, True )
If RetCode <> 1 Then
       ' Display error
       Response.Write(obj.LastError)
Else
       ' Display agent list
       Response.Write(NameList)
End if
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.GetAgentList( true );
if (RetCode != 1)
       // Display error
       Response.Write(obj.LastError);
else
       // Display agent list
       Response.Write(obj.ipStringBuffer);
```

GetAgentProfile

Standard COM and VBScript COM interfaces:
Long GetAgentProfile (String ProfileName,
ByRef String ComMethod,
ByRef String Location,
ByRef String Port,
ByRef String Options)
JavaScript COM:
Long GetAgentProfile (String ProfileName)

The **GetAgentProfile** function retrieves properties of an existing 24x7 Remote Agent profile.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

If the function succeeds, in standard and VBScript COM interfaces all parameters passed by reference are populated with profile properties.

In JavaScript COM interface all profile properties are returned as a single tab-separated string using ipStringBuffer COM object property. You can use GetToken to parse the returned string and extract separate elements.

Parameters:

For description of **GetAgentProfile** parameters see AddAgentProfile method.

See also:

AddAgentProfile
UpdateAgentProfile
DeleteAgentProfile
GetAgentList

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, ComMethod As String, _
       Location As String, Port As String, Options As String

RetCode = obj.GetAgentProfile( "Oracle Agent", _
              ComMethod, Location, Port, Options)
If RetCode <> 1 Then
       ' Display error message
       MsgBox obj.LastError
Else
       ' Display profile properties
       MsgBox "Profile Properties " & vbCrLf & _
              "Profile name: Oracle Agent" & vbCrLf & _
              "Communication Method: " & ComMethod & vbCrLf & _
              "Agent Location: " & Location & vbCrLf & _
              "Port: " & Port & vbCrLf & _
              "Options: " & Options
```

```
End If
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, ComMethod, Location, Port, Options

RetCode = obj.GetAgentProfile( "Oracle Agent", _
             ComMethod, Location, Port, Options)
If RetCode <> 1 Then
       ' Display error message
       Response.Write(obj.LastError)
Else
       ' Display profile properties
       Response.Write("<p>Agent Profile Properties</p>")
       Response.Write("<table>")
       Response.Write("<tr><td>Profile name</td>")
       Response.Write("<td>Oracle Agent</td></tr>")
       Response.Write("<tr><td>Communication Method</td>")
       Response.Write("<td>" & ComMethod & "</td></tr>")
       Response.Write("<tr><td>Agent Location</td>")
       Response.Write("<td>" & Location & "</td></tr>")
       Response.Write("<tr><td>Port</td>")
       Response.Write("<td>" & Port & "</td></tr>")
       Response.Write("<tr><td>Options</td>")
       Response.Write("<td>" & Options & "</td></tr>")
       Response.Write("</table>")
End If
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = 0;

// get profile properties as a tab-separated string
RetCode = obj.GetAgentProfile( "Oracle Agent" );
if (RetCode != 1)
       // Display error
       Response.Write(obj.LastError);
else
{
       // split returned string into array of properties
       var aProperties = obj.ipStringBuffer.split("\t");
       var ComMethod = aProperties[0];
       var Location = aProperties[1];
       var Port = aProperties[2];
       var Options = aProperties[3];
}
```

GetDatabaseList

Standard COM and VBScript COM interfaces:
Long GetDatabaseList (ByRef String Buffer,
Boolean HTMLFormat)

JavaScript COM:
Long GetDatabaseList (Boolean HTMLFormat)

The **GetDatabaseList** function obtains list of names of configured Database Profiles.

If the function succeeds, in standard and VBScript COM interfaces the **Buffer** variable is populated with the list of database profile names. If **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as a HTML table; otherwise it is populated with a plain text containing each entry on a new line.

In JavaScript COM interface the **Buffer** parameter is not used. To obtain the list of database profile names use ipStringBuffer property. If **HTMLFormat** is TRUE, the **ipStringBuffer** property is populated with a text formatted as a HTML table; otherwise it is populated with a plain text containing each entry on a new line.

Note: You can use GetToken function to parse the returned list and extract individual elements or convert it into an array of elements. For details see description and examples for **GetToken** function. In JavaScript you can also use the built-in *split()* function to convert the returned list into an array of elements.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

Buffer This parameter is not available in JavaScript COM interface	A String variable passed by reference. In case of successful function execution, the profile list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the list of Database Profile names. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:

AddDatabaseProfile
DeleteDatabaseProfile
GetDatabaseProfile

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, NameList As String
RetCode = obj.GetDatabaseList( NameList, False )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, NameList
RetCode = obj.GetDatabaseList( NameList, True )
If RetCode <> 1 Then
       ' Display error
       Response.Write(obj.LastError)
Else
       ' Display agent list
       Response.Write(NameList)
End if
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...

var RetCode = obj.GetDatabaseList( true );
if (RetCode != 1)
       // Display error
       Response.Write(obj.LastError);
else
       // Display agent list
       Response.Write(obj.ipStringBuffer);
```

GetDatabaseProfile

Standard COM and VBScript COM interfaces:

Long GetDatabaseProfile (String ProfileName,
ByRef String DatabaseDriver,
ByRef String Server,
ByRef String DatabaseName,
ByRef Boolean AutoCommit,
ByRef String User,
ByRef String Password)

JavaScript COM:

Long GetDatabaseProfile (String ProfileName)

The **GetDatabaseProfile** function retrieves properties of an existing database profile.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

If the function succeeds, in standard and VBScript COM interfaces all parameters passed by reference are populated with profile properties.

In JavaScript COM interface all profile properties are returned as a single tab-separated string using ipStringBuffer COM object property. You can use GetToken to parse the returned string and extract separate property values.

Parameters:

For description of **GetDatabaseProfile** parameters see AddDatabaseProfile method.

See also:

AddDatabaseProfile
UpdateDatabaseProfile
DeleteDatabaseProfile

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, DatabaseDriver As String, _
      Server As String, DatabaseName As String, _
      AutoCommit As Boolean, User As String, Password As String

RetCode = obj.GetDatabaseProfile( "Prod Server", _
            DatabaseDriver, Server, DatabaseName, _
            AutoCommit, User, Password)
If RetCode <> 1 Then
      ' Display error message
      MsgBox obj.LastError
Else
      ' Display profile properties
      MsgBox "Profile Properties " & vbCrLf & _
            "Profile name: Prod Server" & vbCrLf & _
            "Driver: " & DatabaseDriver & vbCrLf & _
```

```
                "Server Name: " & Server & vbCrLf & _
                "Database Name: " & DatabaseName & vbCrLf & _
                "AutoCommit Mode: " & CStr(AutoCommit) & vbCrLf & _
                "User: " & User & vbCrLf & _
                "Password: " & Password
End If
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, DatabaseDriver, Server, DatabaseName, _
       AutoCommit, User, Password

RetCode = obj.GetDatabaseProfile( "Prod Server", _
              DatabaseDriver, Server, DatabaseName, _
              AutoCommit, User, Password)
If RetCode <> 1 Then
       ' Display error message
       Response.Write(obj.LastError)
Else
       ' Display profile properties
       Response.Write("<p>Profile Properties</p>")
       Response.Write("<table>")
       Response.Write("<tr><td>Profile name</td>")
       Response.Write("<td>Prod Server</td></tr>")
       Response.Write("<tr><td>Driver</td>")
       Response.Write("<td>" & DatabaseDriver & "</td></tr>")
       Response.Write("<tr><td>Server Name</td>")
       Response.Write("<td>" & Server & "</td></tr>")
       Response.Write("<tr><td>Database Name</td>")
       Response.Write("<td>" & DatabaseName & "</td></tr>")
       Response.Write("<tr><td>AutoCommit Mode</td>")
       Response.Write("<td>" & CStr(AutoCommit) & "</td></tr>")
       Response.Write("<tr><td>User</td>")
       Response.Write("<td>" & User & "</td></tr>")
       Response.Write("<tr><td>Password</td>")
       Response.Write("<td>" & Password & "</td></tr>")
       Response.Write("</table>")
End If
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = 0;

// get profile properties as a tab-separated string
RetCode = obj.GetDatabaseProfile( "Prod Server" );
if (RetCode != 1)
       // Display error
       Response.Write(obj.LastError);
else
{
       // split returned string into array of properties
       var aProperties = obj.ipStringBuffer.split("\t");
       var DatabaseDriver = aProperties[0];
       var Server = aProperties[1];
       var DatabaseName = aProperties[2];
```

```
        var AutoCommit = (aProperties[3] == "true");
        var User  = aProperties[4];
        var Password  = aProperties[5];
    }
```

GetFolderList

Standard COM and VBScript COM interfaces:
Long GetFolderList (ByRef String Buffer,
Boolean HTMLFormat)

JavaScript COM:
Long GetFolderList (Boolean HTMLFormat)

The **GetFolderList** function obtains list of all job folders.

If the function succeeds, in standard and VBScript COM interfaces the **Buffer** variable is populated with the list of job folder IDs and names. If **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as a HTML table; otherwise it is populated with a plain text containing each entry on a new line. Each line contains folder ID and name separated by a tab character.

In JavaScript COM interface the **Buffer** parameter is not used. To obtain the list of folders IDs and names use ipStringBuffer property. If **HTMLFormat** is TRUE, the **ipStringBuffer** property is populated with a text formatted as a HTML table; otherwise it is populated with a plain text containing each entry on a new line. Each line contains folder ID and name separated by a tab character.

Note: You can use GetToken function to parse the returned list and extract individual elements or convert it into an array of elements. For details see description and examples for **GetToken** function. In JavaScript you can also use the built-in `split()` function to convert the returned list into an array of elements.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

Buffer This parameter is not available in JavaScript COM interface	A String variable passed by reference. In case of successful function execution, the folder list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the result. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:

CreateFolder
DeleteFolder
SetFolderProperty
GetFolderProperty

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, Folders As String
RetCode = obj.GetFolderList( Folders, False )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, Folders
RetCode = obj.GetFolderList( Folders, True )
If RetCode <> 1 Then
       ' Display error
       Response.Write(obj.LastError)
Else
       ' Display folder list
       Response.Write(Folders)
End if
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...

var RetCode = obj.GetFolderList( true );
if (RetCode != 1)
       // Display error
       Response.Write(obj.LastError);
else
       // Display folder list
       Response.Write(obj.ipStringBuffer);
```

GetFolderProperty

Standard COM and VBScript COM interfaces:
Long GetFolderProperty (String FolderID,
String PropertyName,
ByRef String Buffer)

JavaScript COM:
Long GetFolderProperty (String FolderID,
String PropertyName)

The **GetFolderProperty** function obtains value of the specified job **PropertyName** for the specified **JobID**. The **PropertyName** must be a valid JDL job property name. The following folder properties are supported: "FOLDER_NAME", "DESCRIPTION", "FOLDER", "MODIFY_TIME", "MODIFY_USER", "MODIFY_TERMINAL"

If the function succeeds, in standard and VBScript COM interfaces the **Buffer** variable is populated with the value of the folder property.

In JavaScript COM interface the **Buffer** parameter is not used. The value of requested folder property is returned using ipStringBuffer COM object property.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

FolderID	Folder ID converted to string or Folder name.
PropertyName	The folder property name.
Buffer This parameter is not available in JavaScript COM interface	A String variable passed by reference. In case of successful function execution, the property value is copied to the **Buffer**.

See also:
CreateFolder
SetFolderProperty

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, Name As String
' Retrieve name of folder whose ID is -12
RetCode = obj.GetFolderPropery( "-12", "FOLDER_NAME", Name )
```

```
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, Name
' Retrieve name of folder whose ID is -12
RetCode = obj.GetFolderPropery( "-12", "FOLDER_NAME", Name )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...

var RetCode = obj.GetFolderProperty( "-12", "FOLDER_NAME" );
if (RetCode != 1)
	// Display error
	Response.Write(obj.LastError);
else
	// copy folder name
	var Name = obj.ipStringBuffer;
```

GetForecast

Standard COM and VBScript COM interfaces:
Long GetForecast (ByRef String Buffer,
Boolean HTMLFormat)

JavaScript COM:
Long GetForecast (Boolean HTMLFormat)

The **GetForecast** function obtains 7-days job forecast.

If the function succeeds, in standard and VBScript COM interfaces the **Buffer** variable is populated with job forecast report. If **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as a HTML table; otherwise it is populated with a plain text containing each entry on a new line.

In JavaScript COM interface the **Buffer** parameter is not used. To obtain the forecast use ipStringBuffer property. If **HTMLFormat** is TRUE, the **ipStringBuffer** property is populated with a text formatted as a HTML table; otherwise it is populated with a plain text containing each entry on a new line.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

Buffer This parameter is not available in JavaScript COM interface	A String variable passed by reference. In case of successful function execution, the job list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the list of forecasted jobs. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:
GetMonitor
GetJobQueueMonitor

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, Report As String
RetCode = obj.GetForecast( Report, False )
If RetCode <> 1 Then
	MsgBox obj.LastError ' Display error message
Else
	Form1.Report.Text = Report  ' Display job Forecast
End If
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, Report
RetCode = obj.GetForecast( Report, True )
If RetCode <> 1 Then
	Response.Write(obj.LastError) ' Display error
Else
	Response.Write(Report) ' Display report
End If
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...

var RetCode = obj.GetForecast( true );
if (RetCode != 1)
	// Display error
	Response.Write(obj.LastError);
else
	// Display report
	Response.Write(obj.ipStringBuffer);
```

GetGlobalVariable

Standard COM and VBScript COM interfaces:
Long GetGlobalVariable (String VariableName,
ByRef String Buffer)

JavaScript COM:
Long GetGlobalVariable (String VariableName)

The **GetGlobalVariable** function obtains value of the global variable on the target 24x7 Remote Agent or 24x7 Master Scheduler.

If the function succeeds, in standard and VBScript COM interfaces the **Buffer** variable is populated with the returned value.

Important Notes:

- This method is only valid for 24x7 Scheduler Windows Edition.
- In JavaScript COM interface the **Buffer** parameter is not used. To obtain the returned value use ipStringBuffer property.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

VariableName	The name of the global variable that must exist on the target 24x7 Agent or Master Scheduler. Variable names are case-insensitive.
Buffer This parameter is not available in JavaScript COM interface	A String variable passed by reference. In case of successful function execution, the value of the specified global variable is copied to the **Buffer**.

See also:

SetGlobalVariable
RunScript

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, VarValue As String
RetCode = obj.GetGlobalVariable( "my_variable", VarValue )
If RetCode <> 1 Then
```

```
	MsgBox obj.LastError ' Display error message
Else
	MsgBox VarValue  ' Display the returned global value
End If
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, VarValue
RetCode = obj.GetGlobalVariable( "my_variable", VarValue )
If RetCode <> 1 Then
	Response.Write(obj.LastError) ' Display error
Else
	Response.Write(VarValue) ' Display the returned global value
End If
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...

var RetCode = 0;
RetCode = obj.GetGlobalVariable( "my_variable" );
if (RetCode != 1)
{
	// Display error
	Response.Write(obj.LastError);
}
else
{
	// Display the returned global value
	var VarValue = obj.ipStringBuffer;
	Response.Write(VarValue);
}
```

GetHolidays

Standard COM and VBScript COM interfaces:
Long GetHolidays (Long Year,
ByRef String Buffer,
Boolean HTMLFormat)

JavaScript COM:
Long GetHolidays (Long Year,
Boolean HTMLFormat)

The **GetHolidays** function obtains list of holidays for the specified **Year**.

If the function succeeds, in standard and VBScript COM interfaces the **Buffer** variable is populated with the list of holidays. If **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as a HTML table; otherwise it is populated with a plain text containing each entry on a new line. Each line contains holiday date and description separated by tab characters.

In JavaScript COM interface the **Buffer** parameter is not used. Use the ipStringBuffer COM object property to obtain the returned holiday list. If **HTMLFormat** is TRUE, the **ipStringBuffer** property is populated with a text formatted as a HTML table; otherwise it is populated with a plain text containing each entry on a new line. Each line contains holiday date and description separated by tab characters.

Note: You can use GetToken function to parse the returned list and extract individual elements or convert it into an array of elements. For details see description and examples for **GetToken** function. In JavaScript you can also use the built-in *split()* function to convert the returned list into an array of elements.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

Year	The year for which you want to obtain the list of holidays.
Buffer This parameter is not available in JavaScript COM interface	A String variable passed by reference. In case of successful function execution, the holiday list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the list of Holidays. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:

AddHoliday
DeleteHoliday

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, Holidays As String
RetCode = obj.GetHolidays( 2003, Holidays, False )
If RetCode <> 1 Then
       ' Display error message
       MsgBox obj.LastError
Else
       ' Display holidays
       Form1.Holidays.Text = Holidays
End If
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, Holidays
RetCode = obj.GetHolidays( 2003, Holidays, True )
If RetCode <> 1 Then
       ' Display error
       Response.Write(obj.LastError)
Else
       ' Display holidays
       Response.Write(Holidays)
End if
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...

var RetCode = obj.GetHolidays( 2003, true );
if (RetCode != 1)
       // Display error
       Response.Write(obj.LastError);
else
       // Display holidays
       Response.Write(obj.ipStringBuffer);
```

GetJobDefinition

Standard COM and VBScript COM interfaces:
Long GetJobDefinition (String JobID,
ByRef String Buffer,
Boolean HTMLFormat)

JavaScript COM:
Long GetJobDefinition (String JobID,
Boolean HTMLFormat)

The **GetJobDefinition** function obtains values of all job properties for the specified **JobID**.

If the function succeeds, in standard and VBScript COM interfaces the **Buffer** variable is populated with the job definition. If **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as an HTML table. When displayed in a Web browser the table looks like the Job Properties View presented in the 24x7 GUI. If **HTMLFormat** is FALSE, the **Buffer** variable is populated with a plain text containing all job properties in JDL file format. The returned data is compatible with the CreateJob function.

In JavaScript COM interface the **Buffer** parameter is not used. Use the ipStringBuffer COM object property to obtain the returned data. If **HTMLFormat** is TRUE, the **ipStringBuffer** property is populated with a text formatted as a HTML table.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name.
Buffer This parameter is not available in JavaScript COM interface	A String variable passed by reference. In case of successful function execution, the job properties are copied to the **Buffer**. For supported job properties and their JDL names see 24x7 User's Guide. As a quick reference see Job Properties in JDL Format topic in this manual.
HTMLFormat	The output format in which you want to obtain the result. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:

CreateJob
GetJobProperty
GetFolderProperty
GetJobTemplateData
UpdateJob

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, JobDef As String
' Retrieve properties of job whose ID is 245
RetCode = obj.GetJobDefinition( "245", JobDef, False )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, JobDef
' Retrieve properties of job whose ID is 245
RetCode = obj.GetJobDefinition( "245", JobDef, True )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...

var RetCode = obj.GetJobDefinition( "245", true );
if (RetCode != 1)
       // Display error
       Response.Write(obj.LastError);
else
       // copy returned definition
       var JobDef = obj.ipStringBuffer;
```

GetJobList

Standard COM and VBScript COM interfaces:
Long GetJobList (ByRef String Buffer,
Boolean HTMLFormat)

JavaScript COM:
Long GetJobList (Boolean HTMLFormat)

The **GetJobList** function obtains list of all jobs.

If the **HTMLFormat** is TRUE, in standard and VBScript COM interfaces the **Buffer** variable is populated with a text formatted as an HTML table; otherwise the **Buffer** variable is populated with a plain text containing all job IDs and names. Each job appears on a new line. Job IDs and names are separated by tab characters.

In JavaScript COM interface the **Buffer** parameter is not used. Use the ipStringBuffer COM object property to obtain the returned data. If **HTMLFormat** is TRUE, the **ipStringBuffer** property is populated with a text formatted as a HTML table.

Note: You can use GetToken function to parse the returned list and extract individual elements or convert it into an array of elements. For details see description and examples for **GetToken** function. In JavaScript you can also use the built-in *split()* function to convert the returned list into an array of elements.

Sample output (with **HTMLFormat** set to FALSE):
1 Test job 1
5 Backup job
4 Database export
27 Database replication

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

Buffer This parameter is not available in JavaScript COM interface	A String variable passed by reference. In case of successful function execution, the job list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the result. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:

GetJobListEx
GetFolderList

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, JobList As String
' Obtain job list
RetCode = obj.GetJobList( JobList, False )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, JobList
' Obtain job list as HTML table
RetCode = obj.GetJobList( JobList, True )
If RetCode <> 1 Then
	Response.Write(obj.LastError) ' Display error
Else
	Response.Write(JobList) ' Display job list
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
// Obtain job list as HTML table
var RetCode = obj.GetJobList( true );
if (RetCode != 1)
{
	// Display error
	Response.Write(obj.LastError);
}
else
{
	// copy returned job list
	var JobList = obj.ipStringBuffer;
	// Display job list
	Response.Write(JobList);
}
```

GetJobListEx

Standard COM and VBScript COM interfaces:
Long GetJobListEx (ByRef String Buffer,
Boolean HTMLFormat,
String PropertyList)

JavaScript COM:
Long GetJobListEx (Boolean HTMLFormat,
String PropertyList)

The **GetJobListEx** function obtains list of all jobs and optionally their properties.

The **GetJobListEx** is an extended version of GetJobList function. **GetJobListEx** allows specifying additional job properties for inclusion in the returned job list. This generally provides much better performance as compared to first calling **GetJobList** and then calling GetJobProperty function for every returned job in order to obtain job properties.

Job ID and Job Name properties are always included in the returned list that is why they need not be specified in the **PropertyList** parameter. Use **PropertyList** parameter to specify additional JDL properties. Separate multiple properties by commas. For example: "FOLDER_NAME,DISABLED,MODIFY_TIME".

If the **HTMLFormat** is TRUE, in standard and VBScript COM interfaces the **Buffer** variable is populated with a text formatted as an HTML table; otherwise the **Buffer** variable is populated with a plain text containing all job IDs and names and other job parameters specified by **PropertyList** . Each job appears on a new line. Job IDs, names and additional properties are separated by tab characters.

In JavaScript COM interface the **Buffer** parameter is not used. Use the ipStringBuffer COM object property to obtain the returned data. If **HTMLFormat** is TRUE, the **ipStringBuffer** property is populated with a text formatted as a HTML table.

Note: You can use GetToken function to parse the returned list and extract individual elements or convert it into an array of elements. For details see description and examples for **GetToken** function. In JavaScript you can also use the built-in `split()` function to convert the returned list into an array of elements.

Sample output (with **HTMLFormat** set to FALSE and **PropertyList** parameter set to "FOLDER_NAME,DISABLED"):

1	Test job 1	Backup jobs folder	Y
5	Backup job	Backup jobs folder	N
4	Database export	Database jobs	N
27	Database replication	Database jobs	N

If the **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as an HTML table; otherwise the **Buffer** variable is populated with a plain text containing all job IDs and names. Each job appears on a new line. Job IDs, names, and additional properties specified by **PropertyList** are separated by tab characters.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

Buffer This parameter is not available in JavaScript COM interface	A String variable passed by reference. In case of successful function execution, the job list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the result. **Value** **Meaning** True format output as HTML text False format output as plain text
PropertyList	A comma separated list of job property names. For supported job properties and their JDL names see 24x7 User's Guide. As a quick reference see Job Properties in JDL Format topic in this manual.

See also:

GetJobList
GetFolderList
GetJobProperty
GetJobPropertyEx

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, JobList As String
' Obtain job list
RetCode = obj.GetJobListEx( JobList, False, "JOB_TYPE,DISABLED" )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, JobList
' Obtain job list as HTML table
RetCode = obj.GetJobListEx( JobList, True, "JOB_TYPE,DISABLED" )
If RetCode <> 1 Then
      Response.Write(obj.LastError) ' Display error
Else
      Response.Write(JobList) ' Display job list
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
// Obtain job list as HTML table
var RetCode = obj.GetJobListEx( True, "JOB_TYPE,DISABLED" );
if (RetCode != 1)
{
	// Display error
	Response.Write(obj.LastError);
}
else
{
	// copy returned job list
	var JobList = obj.ipStringBuffer;
	// Display job list
	Response.Write(JobList);
}
```

GetJobLog

Standard COM and VBScript COM interfaces:
Long GetJobLog (String JobID,
ByRef String Buffer,
Boolean HTMLFormat)

JavaScript COM:
Long GetJobLog (String JobID,
Boolean HTMLFormat)

The **GetJobLog** function obtains all log records for the specified **JobID**.

If the **HTMLFormat** is TRUE, in standard and VBScript COM interfaces the **Buffer** variable is populated with a text formatted as an HTML table. When displayed in a Web browser the table looks like the Job Log in the 24x7 GUI. If **HTMLFormat** is FALSE, the **Buffer** variable is populated with a plain text containing log records. Each record appears on a new line with columns separated by tab characters. The format of the text is the same as format of the job log file SCHEDULE.LOG.

In JavaScript COM interface the **Buffer** parameter is not used. Use the ipStringBuffer COM object property to obtain the returned data. If **HTMLFormat** is TRUE, the **ipStringBuffer** property is populated with a text formatted as a HTML table.

Note: To obtain the entire log for all jobs specify "0" for the **JobID** parameter.

Note: You can use GetToken function to parse the returned list and extract individual elements or convert it into an array of elements. For details see description and examples for **GetToken** function. In JavaScript you can also use the built-in *`split()`* function to convert the returned list into an array of elements.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name.
Buffer This parameter is not available in JavaScript COM interface	A String variable passed by reference. In case of successful function execution, the job log is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the result. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:

GetForecast
GetMonitor

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, JobLong As String
' Obtain job log for job #285
RetCode = obj.GetJobLog( "285", JobLog, False )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, JobLog
' Obtain job log for job #285 as HTML table
RetCode = obj.GetJobLog( "285", JobLog, True )
If RetCode <> 1 Then
	Response.Write(obj.LastError) ' Display error
Else
	Response.Write(JobLog) ' Display log
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
// Obtain job log for job #285 as HTML table
var RetCode = obj.GetJobLog( "285", true );
if (RetCode != 1)
{
	// Display error
	Response.Write(obj.LastError);
}
else
{
	// copy returned log
	var JobLog = obj.ipStringBuffer;
	// Display log
	Response.Write(JobLog);
}
```

GetJobQueueSize

Standard COM and VBScript COM interfaces:
Long GetJobQueueSize (String QueueName,
ByRef Long MaxSize)
JavaScript COM:
Long GetJobQueueSize (String QueueName)

The **GetJobQueueSize** function retrieves maximum size of an existing job queue.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

If the function succeeds, in standard and VBScript COM interfaces **MaxSize** parameter is populated with job queue size value.

In JavaScript COM interface job queue size value is returned using ipLongBuffer COM object property.

Parameters:

For description of **GetJobQueue** parameters see AddJobQueue method.

See also:

AddJobQueue
UpdateJobQueue
DeleteJobQueue

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, MaxSize As Long

RetCode = obj.GetJobQueueSize( "Weekly jobs", MaxSize)
If RetCode <> 1 Then
       ' Display error message
       MsgBox obj.LastError
Else
       ' Display queue properties
       MsgBox "Queue Properties " & vbCrLf & _
              "Queue name: Weekly jobs" & vbCrLf & _
              "Max Queue Size: " & CStr(MaxSize) & " MB"
End If
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, MaxSize

RetCode = obj.GetJobQueueSize( "Weekly jobs", MaxSize)
```

```
If RetCode <> 1 Then
	' Display error message
	Response.Write(obj.LastError)
Else
	' Display queue properties
	Response.Write("<p>Queue Properties</p>")
	Response.Write("<table>")
	Response.Write("<tr><td>Queue name</td>")
	Response.Write("<td>Weekly jobs</td></tr>")
	Response.Write("<tr><td>Max Queue Size</td>")
	Response.Write("<td>" & CStr(MaxSize) & " MB</td></tr>")
	Response.Write("</table>")
End If
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = 0;

// get queue properties
RetCode = obj.GetJobQueueSize( "Weekly jobs" );
if (RetCode != 1)
{
	// Display error
	Response.Write(obj.LastError);
}
else
{
	// copy job queue size value
	var MaxSize = obj.ipLongBuffer;

	// Display queue properties
	Response.Write("<p>Queue Properties</p>");
	Response.Write("<table>");
	Response.Write("<tr><td>Queue name</td>");
	Response.Write("<td>Weekly jobs</td></tr>");
	Response.Write("<tr><td>Max Queue Size</td>");
	Response.Write("<td>" + MaxSize.toString() + " MB</td></tr>");
	Response.Write("</table>");

}
```

GetJobQueueList

Standard COM and VBScript COM interfaces:
Long GetJobQueueList(ByRef String Buffer,
Boolean HTMLFormat)

JavaScript COM:
Long GetJobQueueList(Boolean HTMLFormat)

The **GetJobQueueList** function obtains list of names of configured Job Queues.

If **HTMLFormat** is TRUE, in standard and VBScript COM interfaces the **Buffer** variable is populated with a text formatted as a HTML table; otherwise it is populated with a plain text containing each entry on a new line.

In JavaScript COM interface the **Buffer** parameter is not used. Use the ipStringBuffer COM object property to obtain the returned data. If **HTMLFormat** is TRUE, the **ipStringBuffer** property is populated with a text formatted as a HTML table.

Note: You can use GetToken function to parse the returned list and extract individual elements or convert it into an array of elements. For details see description and examples for **GetToken** function. In JavaScript you can also use the built-in *split()* function to convert the returned list into an array of elements.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

Buffer This parameter is not available in JavaScript COM interface	A String variable passed by reference. In case of successful function execution, the queue list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the list of 24x7 Job Queue names. **Value** **Meaning** True format output as HTML text False format output as plain text

See also:

GetJobQueue
GetJobQueueMonitor
AddJobQueue

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, QList As String
RetCode = obj.GetJobQueueList( QList, False )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, QList
RetCode = obj.GetJobQueueList( QList, False )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.GetJobQueueList( false );
if (RetCode != 1)
	Response.Write(obj.LastError);  // Display error
else
	var QList = obj.ipStringBuffer; // copy returned list
```

GetJobQueueMonitor

Standard COM and VBScript COM interfaces:
Long GetJobQueueMonitor (String QueueName,
ByRef String Buffer,
Boolean HTMLFormat)

JavaScript COM:
Long GetJobQueueMonitor (String QueueName,
Boolean HTMLFormat)

The **GetJobQueueMonitor** function obtains list of running and waiting jobs in the specified job queue. If **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as a HTML table; otherwise it is populated with a plain text containing each entry on a new line. Each line contains the following tab-separated columns: queue name, unique job run number for that queue, job submition date and time, job priority (High, Normal or Low), job status (Awaiting, Running or Held), job ID, job name, job start time (for running jobs only), size of uncompressed job definition and linked job deployement data, size of compressed job definition and linked job deployement data, compression ratio. The same columns are available in the graphical version of the job queue monitor.

In JavaScript COM interface the **Buffer** parameter is not used. Use the ipStringBuffer COM object property to obtain the returned data. If **HTMLFormat** is TRUE, the **ipStringBuffer** property is populated with a text formatted as a HTML table.

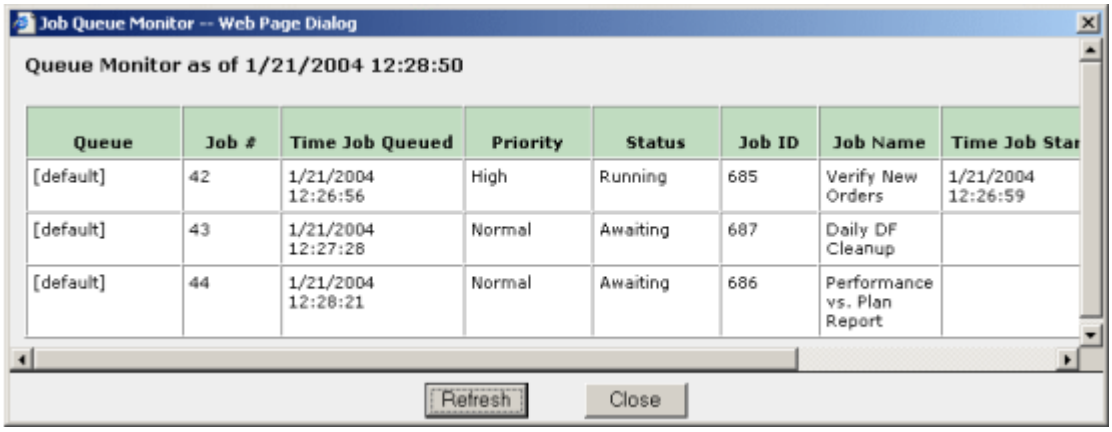
Note: You can use GetToken function to parse the returned list and extract individual elements or convert it into an array of elements. For details see description and examples for **GetToken** function. In JavaScript you can also use the built-in *`split()`* function to convert the returned list into an array of elements.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

QueueName	The job queue name.
Buffer This parameter is not available in JavaScript COM interface	A String variable passed by reference. In case of successful function execution, the job monitor data is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the list of forecasted jobs. **Value** **Meaning** True format output as HTML text False format output as plain text

Sample queue monitor screenshot



See also:

GetForecast
GetMonitor

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, JobList As String
RetCode = obj.GetJobQueueMonitor( "Weekly Jobs", JobList, False )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, MonitorData
RetCode = obj.GetJobQueueMonitor( "Weekly Jobs", MonitorData, True )
If RetCode <> 1 Then
       Response.Write(obj.LastError) ' Display error
Else
       Response.Write(MonitorData) ' Display monitor data
                                   ' as HTML table
End If
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.GetJobQueueMonitor( "Weekly Jobs", true );
if (RetCode != 1)
       Response.Write(obj.LastError);  // Display error
else
       Response.Write(obj.ipStringBuffer); // Display monitor data
                                           // as HTML table
```

GetJobProperty

Standard COM and VBScript COM interfaces:
Long GetJobProperty (String JobID,
String PropertyName,
ByRef String Buffer)

JavaScript COM:
Long GetJobProperty (String JobID,
String PropertyName)

The **GetJobProperty** function obtains value of the specified job property. The **PropertyName** must be a valid JDL job property name.

In JavaScript COM interface the **Buffer** parameter is not used. Use the ipStringBuffer COM object property to obtain the returned data.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name. Specify ID or name of the job whose property your want to retrieve.
PropertyName	The job property name. Specify name of the property whose value your want to retrieve. For supported job properties and their JDL names see 24x7 User's Guide. As a quick reference see Job Properties in JDL Format topic in this manual.
Buffer This parameter is not available in JavaScript COM interface	A String variable passed by reference. In case of successful function execution, the property value is copied to the **Buffer**.

See also:

SetJobProperty
GetJobPropertyEx
GetJobDefinition
GetJobTemplateData

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, JobQueue As String
' Retrieve value of QUEUE property for job #235
RetCode = obj.GetJobProperty( "235", "QUEUE", JobQueue )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, JobQueue
' Retrieve value of QUEUE property for job #235
RetCode = obj.GetJobProperty( "235", "QUEUE", JobQueue )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
// Retrieve value of QUEUE property for job #235
var RetCode = GetJobProperty( "235", "QUEUE" );
if (RetCode != 1)
       Response.Write(obj.LastError);  // Display error
else
       var QueueName = obj.ipStringBuffer; // copy returned value
```

GetJobPropertyEx

Standard COM and VBScript COM interfaces:
Long GetJobPropertyEx (String JobID,
String PropertyNameList,
ByRef String Buffer)

JavaScript COM:
Long GetJobPropertyEx (String JobID,
String PropertyNameList)

The **GetJobPropertyEx** function obtains values of one or more job properties. The **PropertyNameList** must contain comma-separated list of valid JDL job property names.

Note: GetJobPropertyEx function is an extended version of GetJobProperty function. **GetJobPropertyEx** is capable of returning values of multiple properties at once while **GetJobProperty** can return only one property at a time. **GetJobPropertyEx** is a more efficient and faster method to get multiple properties as it makes just one pass to the 24x7 Scheduler server.

In JavaScript COM interface the **Buffer** parameter is not used. Use the ipStringBuffer COM object property to obtain the returned data.

Note: You can use GetToken function to parse the returned list and extract individual elements or convert it into an array of elements. For details see description and examples for **GetToken** function. In JavaScript you can also use the built-in *`split()`* function to convert the returned list into an array of elements.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name. . Specify ID or name of the job whose properties your want to retrieve.
PropertyNameList	Comma-separated list of job property names. Specify names of properties whose values your want to retrieve. For supported job properties and their JDL names see 24x7 User's Guide. As a quick reference see Job Properties in JDL Format topic in this manual.
Buffer This parameter is not available in JavaScript COM interface	A String variable passed by reference. In case of successful function execution, property values are copied to the **Buffer** as a tab delimited list.

See also:

GetJobProperty
GetJobDefinition
SetJobProperty
SetJobPropertyEx
GetJobTemplateData

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, ValueList As String, FileName As String
Dim DirName As String, Suffix As String
' Retrieve values of MSG_FILE, MOVE_DIR, RENAME_SUFFIX properties
' for job #235
RetCode = obj.GetJobPropertyEx( "235", _
       "MSG_FILE,MOVE_DIR,RENAME_SUFFIX", ValueList )
If RetCode <> 1 Then
       MsgBox obj.LastError       ' Display error message
Else
       ' Break the returned value into individual property values
       FileName = obj.GetToken(Values, vbTab)
       DirName = obj.GetToken(Values, vbTab)
       Suffix = Values
End If
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, ValueList, FileName, DirName, Suffix
' Retrieve values of MSG_FILE, MOVE_DIR, RENAME_SUFFIX properties
' for job #235
RetCode = obj.GetJobPropertyEx( "235", _
       "MSG_FILE,MOVE_DIR,RENAME_SUFFIX", ValueList )
If RetCode <> 1 Then
       MsgBox obj.LastError       ' Display error message
Else
       ' Break the returned value into individual property values
       FileName = obj.GetToken(Values, vbTab)
       DirName = obj.GetToken(Values, vbTab)
       Suffix = Values
End If
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
// Retrieve values of MSG_FILE, MOVE_DIR, RENAME_SUFFIX properties
// for job #235
var RetCode = obj.GetJobPropertyEx( "235",
       "MSG_FILE,MOVE_DIR,RENAME_SUFFIX" );
if (RetCode != 1)
       Response.Write(obj.LastError);  // Display error
else
{
       // Break the returned value into individual property values
```

```
        var FileName = obj.GetToken("\t");
        var DirName  = obj.GetToken("\t");
        var Suffix = obj.ipStringBuffer;
}
```

GetJobStatus

Long GetJobStatus (String JobID)

The **GetJobStatus** function obtains the status of the specified job.

Return: Returns a number indicating job status, which could be one of the following:

-1	Error job. Job may have this status because of incomplete job definition or if an error occurred while 24x7 was executing this job.
-2	Job is performing notification action after or before job run, such as sending email message, executing database command, etc.
-3	Job is running.
-5	First job run is pending. This job has been never been started before.
-6	Unknown status. Job may have an unknown status when it is a "Program" type job and the operation system is unable to report exit code of the job process.
0	Successfully finished. Note: an asynchronous job of "Program" type also may have this status after it successfully started the specified program and the started program is still running.
other	Successfully finished (valid for jobs of "Program" type only).

Parameters:

JobID	Job ID converted to string or job name.

See also:

RunJob
GetForecast
GetJobQueueMonitor
GetMonitor

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, JobStatus As Long
' Run job #235 in the background
RetCode = obj.RunJob( "235", True )

' ... do something here ...

' Get job status
```

```
JobStatus = obj.GetJobStatus( "235")
If JobStatus >= 0 Then
       MsgBox "Job #235 completed successfully"
ElseIf JobStatus = -2 or JobStatus = -3 Then
       MsgBox "Job #235 completed successfully"
Else
       MsgBox "Job #235 failed. Check the job log for details"
End If
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, JobStatus
' Run job #235 in the background
RetCode = obj.RunJob( "235", True )

' ... do something here ...

' Get job status
JobStatus = obj.GetJobStatus( "235")
If JobStatus >= 0 Then
       Response.Write("Job #235 completed successfully")
ElseIf JobStatus = -2 or JobStatus = -3 Then
       Response.Write("Job #235 completed successfully")
Else
       Response.Write("Job #235 failed. Check the job log for " & _
                             "details")
End If
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
// Run job #235 in the background
var RetCode = obj.RunJob( "235", true );

// ... do something here ...

// Get job status
var JobStatus = obj.GetJobStatus( "235");
if (JobStatus >= 0)
       Response.Write("Job #235 completed successfully");
else if (JobStatus == -2 or JobStatus == -3)
       Response.Write("Job #235 completed successfully");
else
       Response.Write("Job #235 failed. Check the job log for " +
                      "details");
```

GetJobTemplateData

Standard COM and VBScript COM interfaces:
Long GetJobTemplateData (String JobID,
ByRef String TemplateName,
ByRef String TemplateFile,
ByRef String Data)

JavaScript COM:
Long GetJobTemplateData (String JobID)

The **GetJobTemplateData** function obtains template properties and data used to create or modify the specified job. This data must be previously saved using SetJobTemplateData function.

If the **GetJobTemplateData** function succeeds in standard and VBScript COM interfaces the **TemplateName** parameter returns name of the original job template, the **TemplateFile** parameter returns name the original job template file and the **Data** parameter returns the saved data.

In JavaScript COM interface the **TemplateName**, **TemplateFile** and **Data** parameter are not used. Use the ipStringBuffer COM object property to obtain the returned values. If the **GetJobTemplateData** function succeeds the **ipStringBuffer** property is populated with a multi-line text. The first line contans the **TemplateName** value. The second line is contains the **TemplateFile** value and the rest starting with the third line contans the **Data** value.

 **Note:** In JavaScript COM interface you can use GetToken function to parse the returned multi-line result and extract individual lines. For details see description and examples for **GetToken** function.

Return: Returns a number indicating function status, which could be one of the following:

1	Success. Both job definition and job template data have been found and all job ID, template name and file, job and template timestamps match exactly.
0	Success with warnings. Both job definition and job template data have been found but either the job or the template have been modified since the data was saved using **SetJobTemplateData** function. It is still should be possible to continue. Use the **LastError** property to obtain the warnings.
A negative number	Failure. An error has occurred during function call or if the template data cannot be found. Use the **LastError** property to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name.
TemplateName	A String variable passed by reference. In case of successful function execution the saved job template name is copied to this parameter.
TemplateFile	A String variable passed by reference. In case of successful function

	execution the saved job template file name is copied to this parameter.
Data This parameter is not available in JavaScript COM interface	A String variable passed by reference. In case of successful function execution saved job data is copied to this parameter.

See also:

SetJobTemplateData
GetJobDefinition
GetJobProperty
GetTemplate

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, TemplateName AsString, TemplateFile As String, _
    TemplateData As String
RetCode = obj.GetJobTemplateData( "235", TemplateName, _
              TemplateFile, TemplateData  )
If RetCode < 0  Then
       ' Display error message
       MsgBox "GetJobTemplateData error: " & obj.LastError
End If
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, TemplateName, TemplateFile, TemplateData
RetCode = obj.GetJobTemplateData( "235", TemplateName, _
              TemplateFile, TemplateData  )
If RetCode < 0  Then
       ' Display error message
       Response.Write( "GetJobTemplateData error: " & obj.LastError )
End If
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.GetJobTemplateData( "235" );
if (RetCode < 0)
       // display error message
       Response.Write( 'GetJobTemplateData error: ' + obj.LastError );
else
{
       // break the returned value into template name, file
       // and data values
       var Lines = new Array[];
       Lines = obj.ipStringBuffer.split( '\r\n' );
       var TemplateName = Lines[0];
       var TemplateFile = Lines[1];
```

```
        var TemplateData = "";
        for (var i=2; i<Lines.length; I++) TemplateData += Lines[I];
}
```

GetMonitor

Standard COM and VBScript COM interfaces:
Long GetMonitor (Long Interval,
ByRef String Buffer,
Boolean HTMLFormat)

JavaScript COM:
Long GetMonitor (Long Interval,
Boolean HTMLFormat)

The **GetMonitor** function obtains list of running and pending jobs for the specified time interval (hours). Jobs having non-time based schedules are not included in the pending jobs portion of the report as the 24x7 Scheduler has now way to know when such jobs start conditions will be satisfied.

If the function succeeds, in standard and VBScript COM interfaces the **Buffer** variable is populated with job monitor report. If **HTMLFormat** is TRUE, the **Buffer** variable is populated with a text formatted as a HTML table; otherwise it is populated with a plain text containing each entry on a new line. Each line contains the following tab-separated columns: job ID, job name, job status, and job last start time.

In JavaScript COM interface the **Buffer** parameter is not used. To obtain the monitor report use ipStringBuffer property. If **HTMLFormat** is TRUE, the **ipStringBuffer** property is populated with a text formatted as a HTML table; otherwise it is populated with a plain text containing each entry on a new line.

Note: You can use GetToken function to parse the returned list and extract individual elements or convert it into an array of elements. For details see description and examples for **GetToken** function. In JavaScript you can also use the built-in *split()* function to convert the returned list into an array of elements.

Sample output (with **HTMLFormat** set to FALSE):

285	Backup job	AWAITING START	10/11/2002 04:20:00 PM
14	BCP	RUNNING	10/11/2002 03:00:00 PM
112	Cube building	RUNNING	10/11/2002 03:10:00 PM
56	Database export	AWAITING START	10/12/2002 02:00:00 AM

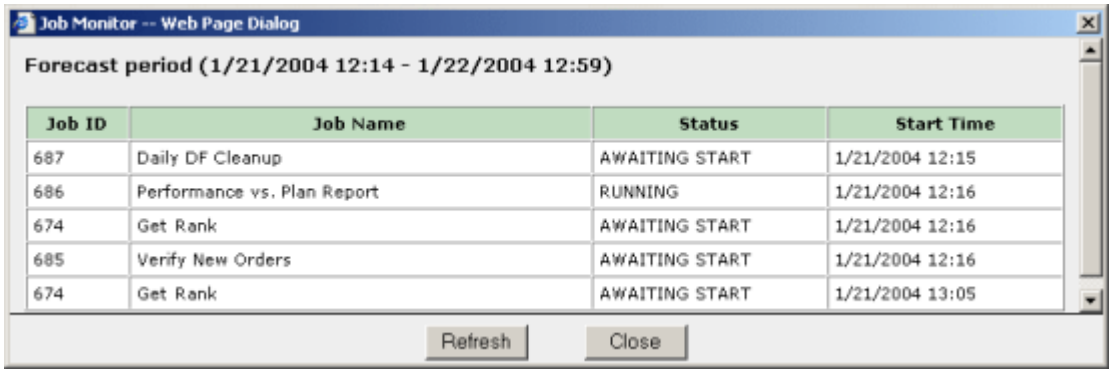
Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

Interval	The time interval within which the monitor report is created. The interval is specified in hours.
Buffer This parameter is not available in JavaScript COM interface	A String variable passed by reference. In case of successful function execution, the job list is copied to the **Buffer**.
HTMLFormat	The output format in which you want to obtain the list of forecasted jobs.

	Value **Meaning** True format output as HTML text False format output as plain text

Sample monitor screenshot



See also:

GetForecast
GetJobQueueMonitor

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, JobList As String
RetCode = obj.GetMonitor( 24, JobList, False )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, MonitorData
RetCode = obj.GetMonitor( 24, MonitorData, True )
If RetCode <> 1 Then
      Response.Write(obj.LastError) ' Display error
Else
      Response.Write(MonitorData) ' Display monitor data
End If
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.GetMonitor( 24, true );
if (RetCode != 1)
      Response.Write(obj.LastError);  // Display error
else
      Response.Write(obj.ipStringBuffer); // Display monitor data
```

GetTemplate

Standard COM and VBScript COM interfaces:
Long GetTemplate (String TemplateFile,
ByRef String Buffer)

JavaScript COM:
Long GetTemplate (String TemplateFile)

The **GetTemplate** function obtains contents of the specified template file.

If the function succeeds, in standard and VBScript COM interfaces the **Buffer** variable is populated with the template text.

In JavaScript COM interface the **Buffer** parameter is not used. To obtain the result use ipStringBuffer property, which is populated with the template text.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

TemplateFile	Template file name.
Buffer This parameter is not available in JavaScript COM interface	A String variable passed by reference. In case of successful function execution, text of the template file is copied to the **Buffer**.

See also:

SetTemplate
GetTemplateCatalog

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, TemplateText As String
RetCode = obj.GetTemplate( "%HOME%\Template\ftp.ini", TemplateText )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, TemplateText
RetCode = obj.GetTemplate( "%HOME%\Template\ftp.ini", TemplateText )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.GetTemplate( '%HOME%\\Template\\ftp.ini',
                               TemplateText );
if (RetCode != 1) Response.Write(obj.LastError);  // Display error
```

GetTemplateCatalog

Standard COM and VBScript COM interfaces:
Long GetTemplateCatalog (ByRef String Buffer)

JavaScript COM:
Long GetTemplateCatalog ( )

The **GetTemplateCatalog** function obtains contents of the TEMPLATE.INI file. TEMPLATE.INI stores catalog of available job templates. The file must exist in the 24x7 Scheduler home directory.

If the function succeeds, in standard and VBScript COM interfaces the **Buffer** variable is populated with the template catalog.

In JavaScript COM interface the **Buffer** parameter is not used. To obtain the result use ipStringBuffer property, which is populated with the template catalog.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

Buffer This parameter is not available in JavaScript COM interface	A String variable passed by reference. In case of successful function execution, contents of the TEMPLATE.INI file is copied to the **Buffer**.

See also:

SetTemplate
GetTemplate
AddTemplate

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, Catalog As String
RetCode = obj.GetTemplateCatalog( Catalog )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, Catalog
RetCode = obj.GetTemplateCatalog( Catalog )
If RetCode <> 1 Then
```

```
        Response.Write(obj.LastError) ' Display error
End If
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.GetTemplateCatalog(  );
if (RetCode != 1)
        Response.Write(obj.LastError);  // Display error
else
        var Catalog = obj.ipStringBuffer; // Copy returned value
```

GetToken

Standard COM and VBScript COM interfaces:
String GetToken (ByRef String Buffer,
String Separator)

JavaScript COM:
String GetToken (String Separator)

The **GetToken** function parses the source string (**Buffer**) and obtains first token. If the token is found it removes it along with the following **Separator** string from the source string.

In JavaScript COM interface the **GetToken** function parses and modifies value of the ipStringBuffer COM object property.

Return: Returns found token or an empty string string if the specified **Separator** value cannot be found or if it is found in the beginning of the **Buffer**.

Parameters:

Buffer This parameter is not available in JavaScript COM interface	A String variable passed by reference. In case of successful function execution, value of the variable pointed by the **Buffer** parameter is modified. The new value does not include the found token and the following separator.
Separator	A string whose value is searched within the source string. The **Separator** value must consist of one or more chacacters.

See also: GetToken can be used with many methods that return lists and and multi-value results, including
GetJobList
GetFolderList
GetJobPropertyEx
GetMonitor
GetJobQueueMonitor
and other.

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, JobList As String
Dim Count As Integer
Dim JobID() As Integer, JobName() As String, JobType() As String
```

```
' Obtain job list using GetJobListEx method (get job IDs, names
' and types)
RetCode = obj.GetJobListEx( JobList, False, "JOB_TYPE" )
If RetCode <> 1 Then
	' Display error message
	MsgBox obj.LastError
Else
	' Parse job list into arrays of job IDs, names and types
	Do While JobList <> ""
		ReDim Preserve JobID(Count)
		ReDim Preserve JobName(Count)
		ReDim Preserve JobType(Count)

		JobID(Count) =   obj.GetToken(JobList, vbTab)
		JobName(Count) = obj.GetToken(JobList, vbTab)
		JobType(Count) = obj.GetToken(JobList, vbCrLf)

		Count = Count + 1
	Loop

	' Display job count
	MsgBox CStr(Count) & " jobs found"
End If
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, JobList, Count
Dim JobID(), JobName(), JobType()

' Obtain job list using GetJobListEx method
RetCode = obj.GetJobListEx( JobList, False, "JOB_TYPE" )
If RetCode <> 1 Then
	' Display error message
	Response.Write(obj.LastError)
Else
	Count = 0

	' Parse job list into arrays of job IDs, names and types
	Do While JobList <> ""
		ReDim Preserve JobID(Count)
		ReDim Preserve JobName(Count)
		ReDim Preserve JobType(Count)

		JobID(Count) =   obj.GetToken(JobList, vbTab)
		JobName(Count) = obj.GetToken(JobList, vbTab)
		JobType(Count) = obj.GetToken(JobList, vbCrLf)

		Count = Count + 1
	Loop

	' Display job count
	Response.Write(CStr(Count) & " jobs found")
End If
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var Count = 0;
```

```
var JobID = new Array();
var JobName = new Array();
var JobType = new Array();

// Obtain job list using GetJobListEx method
var RetCode = obj.GetJobListEx( False, "JOB_TYPE" );
if (RetCode != 1)
{
	// Display error message
	Response.Write(obj.LastError);
}
else
{
	// Parse job list into arrays of job IDs, names and types
	while(obj.ipStringBuffer != "")
	{
		JobID[Count] =   obj.GetToken("\t");
		JobName[Count] = obj.GetToken("\t");
		JobType[Count] = obj.GetToken("\r\n");

		Count ++;
	}

	// Display job count
	Response.Write(Count.toString() + " jobs found");
}
```

GetStatusReport

Long GetStatusReport (String DestinDir)

The **GetStatusReport** function obtains **24x7 Status Report** from the 24x7 Master Scheduler and copies it to the destination directory specified in the **DestinDir** variable.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

DestinDir	The name of the directory into which the **24x7 Status Report** files will be copied. The **DestinDir** can contain either an absolute path to the directory in the format *disk:**dir**subdir1*\...*subdirX* or a relative path in the format *subdir1*\...*sudirX* or a network path name like *SERVER**VOLUME**dir**subdir1*\...*subdirX.* Note: The **DestinDir** must be an existing directory and the directory must be accessible from the computer running 24x7 Scheduler server.

See also:

GetJobLog

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.GetStatusReport( "d:\logs\24x7" )
If RetCode <> 1 Then
       ' Display error message
       MsgBox obj.LastError
Else
       ' Open Status Reports in a Web browser
       Shell "START d:\logs\24x7\index.htm"
End If
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.GetStatusReport( "d:\inetpub\wwwroot\24x7\logs" )
If RetCode <> 1 Then
       ' Display error message
       Response.Write(obj.LastError)
Else
       ' Redirect to Status Reports
       Response.Redirect("..\24x7\logs\index.htm")
End If
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode =
       obj.GetStatusReport( 'd:\\inetpub\\wwwroot\\24x7\\logs' );
if (RetCode != 1)
       // Display error
       Response.Write(obj.LastError);
else
       // Redirect to Status Reports
       Response.Redirect( '..\\24x7\\logs\\index.htm' );
```

GetUserRole

Long GetUserRole (String UserName)

The **GetUserRole** function obtains ID of the security member group (e.g. user role) for the specified user.

Return: The following values can be returned
SECURITY_OFF – the security system is turned off. All users have unlimited privileges.
ROLE_ADMIN – user is a member of Administrators group.
ROLE_STANDARD - user is a member of Standard Privileges group
ROLE_RESTRICTED - user is a member of Restricted Privileges group
ROLE_GUEST - user is a member of Guest group
-1 – an error has occurred. Use the **LastError** property to obtain the error message.

For more information about returned values see Constants topic.

Parameters:

UserName	The name of the user as it is specified in the 24x7 Scheduler security settings. Names are case-insensitive.

See also:
OpenSession

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.GetUserRole( obj.UserID )
If RetCode = -1 Then
       MsgBox obj.LastError ' Display error message
ElseIf RetCode <> obj.ROLE_ADMIN Then
       MsgBox "You are not authorized to use this function"
End If
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.GetUserRole( obj.UserID )
If RetCode = -1 Then
       Response.Write(obj.LastError) ' Display error
ElseIf RetCode <> obj.ROLE_ADMIN Then
       Response.Write("You are not authorized to use this function")
End If
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.GetUserRole( obj.UserID );
if (RetCode == -1)
       Response.Write(obj.LastError); // Display error
else if (RetCode != obj.ROLE_ADMIN)
       Response.Write("You are not authorized to use this function");
```

KillJob

Long HoldJob (Long RunJobID)

The **KillJob** function terminates the specified job and removes it fro mthe queue in which this job is running. The specified job must be already running in the job queue. Job run-time IDs are returned by GetMonitor and by GetJobQueueMonitor functions.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

RunJobID	Job Run-time ID.

See also:

ReleaseJob
HoldJob
GetMonitor
GetJobQueueMonitor

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.HoldJob( 15556 )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.HoldJob( 15556 )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.HoldJob( 15556 );
if (RetCode != 1) Response.Write(obj.LastError); // Display error
```

HoldJob

Long HoldJob (Long RunJobID)

The **HoldJob** function places the specified job on hold. The specified job must be already waiting in the job queue. Job run-time IDs are returned by GetMonitor and by GetJobQueueMonitor functions.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

RunJobID	Job Run-time ID.

See also:

ReleaseJob
KillJob
GetMonitor
GetJobQueueMonitor

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.KillJob( 15556 )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.KillJob( 15556 )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.KillJob( 15556 );
if (RetCode != 1) Response.Write(obj.LastError); // Display error
```

LogMessage

Long LogMessage (Long SourceID,
String SourceName,
Long EventSeverity,
String Message,
Boolean ForceReportUpdate)

The **LogMessage** function adds new record to the 24x7 Scheduler event log. If parallel logging to the Windows NT event log is enabled, **LogMessage** also writes an entry at the end of the Windows NT application event log.

Note: 24x7 Scheduler does not verify SourceID and SourceName values. Normally you should use them to specify job ID and job name that the logged message belongs to.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

SourceID	ID of the job for which you add the new message. To add a job independent message specify 0 for the **SourceID**.
SourceName	Descriptive name of the message source.
EventSeverity	The event severity. It can be one of the following: • RC_ERROR • RC_WARNING • RC_INFO For more information about event severity values see Constants topic.
Message	The message that you want to add to the 24x7 event log.
ForceReportUpdate	A Boolean whose value indicates whether 24x7 Scheduler must perform an immediate update of the **24x7 Status Report** or perform a deferred update. **Value** **Meaning** True update immediately False don't update immediately **Important Notes:** • If the **24x7 Status Report** is not enabled, **ForceReportUpdate** parameter is ignored. • "ForceReportUpdate = True" ensures the report is updated immediately so other people who monitor that report can see this change in a real-time. However, immediate updates have some impact on the performance of the 24x7 Scheduler.

See also:

GetJobLog
GetStatusReport

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.LogMessage( "34", "Some job name", obj.RC_INFO, _
                          "Ready to run job #34", True )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.LogMessage( "34", "Some job name", obj.RC_INFO, _
                                "Ready to run job #34", True )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.LogMessage( '34', 'Some job name', obj.RC_INFO,
                          'Ready to run job #34', true );
if (RetCode != 1) Response.Write(obj.LastError); // Display error
```

OpenSession

Long OpenSession (String User,
String Password,
String ComMethod,
String ServerLocation,
String ServerPort,
String Options,
Boolean Trace)

The **OpenSession** function establishes new connection between **24x7 Remote Control COM** control and the specified target 24x7 Scheduler running in either Master or Agent mode. The target 24x7 Scheduler can run on the same computer or run on another networked computer.

Notes:

- You must open a session using the **OpenSession** function before you call other **24x7 Remote Control COM** functions.
- Use the **CloseSession** function to close the session after you don't need it anymore.

 24x7 COM API supports 3 different interfaces. See Supported COM Interfaces topic for information on which interface should be used with your programming environment.

 If you destroy COM object and a session is still open, the COM object automatically calls **CloseSession**. In web based applications object clean up and destruction are often left to the web server. In such applications you usually do not need to call **CloseSession** explicitly.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

User	The user ID that you want to use for the connection. If an empty string is specified for the **User**, **24x7 Remote Control COM** automatically obtains user ID of the user logged on the computer running **24x7 Remote Control COM**.
Password	Reserved. A String value. In the current version you should use the 24x7 serial number.
ComMethod	Name of the communication driver to be used for the connections to the Agent Values are: • WinSock • NamedPipes
ServerLocation	The location of the Agent. For the **WinSock** driver specify either of the following values: • The IP address (for example, 199.99.99.91)

	• The host name of the remote computer (network computer name in workgroup) • LocalHost (this indicates that the Agent resides on the local machine) For the **NamedPipes** driver specify the location portion of the pipe name. The combination of the Location and Application values forms the pipe name. The pipe name is constructed as follows: \\location\PIPE\application. If no location is specified, a local pipe name is constructed using a dot (.) in the machine name portion.
ServerPort	For the **WinSock** driver specify either of the following: • The port number for the Agent (for example, 1096). Each server application requires a unique port number on the server machine. The port number must be greater than 4096 and less than 65536. • The service name (port alias) for Agent. The service name is an indirect reference to the port number. The mapping of the service name to the port number is specified in the TCP/IP services file. Normally, this file (SERVICES) is located in the Windows directory for Win95/98 and in the C:\WINNT\SYSTEM32\DRIVERS\ETC directory for WinNT. You may need to edit this file manually to add 24x7 Port or Service. For the **NamedPipes** driver specify the application portion of the pipe name. The combination of the Location and Application values forms the pipe name. The pipe name is constructed as follows: \\location\PIPE\application. The Application must be unique for the Location you specify.
Options	Additional communications options. If you specify more than one option, you need to separate the options with commas. For the Local driver this property is ignored. **BufSize=n** Sets the connection buffer size to the value specified. The following options apply to the **WinSock** driver only. **MaxRetry=n** Specifies how many times the Remote Control will try to connect when the Agent's listening port is busy. **NoDelay=1** Specifies that each packet be sent without delay. Corresponds to the TCP_NODELAY option. Setting this option may degrade performance significantly. Do not use this option unless you thoroughly understand its implications. **RawData=1** Specifies that raw data be passed over the network. By default, the WinSock driver obscures the data that is passed over the network. Setting this option to 1 overrides the default behavior. Both the Remote Control and Agent must have the same setting. If there is a discrepancy between the Remote Control and Agent's settings, the

	communication fails. Setting this option to 1 may improve performance slightly.

See also:

CloseSession
Using 24x7 COM API in Your Program
Supported COM Interfaces

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' create COM object
Dim obj As Object
Set obj = CreateObject("24x7 Remote Control")
' open session
obj.OpenSession( "john_doe", "password", "WinSock", _
             "LocalHost", "1096", "", False)
' ... do something here ...
' close session
obj.CloseSession()
' destroy COM object
Set obj = Nothing
```

2. ASP example using VBScript COM interface

```
' create COM object
Dim obj
Set obj = Server.CreateObject("w24x7ASP.vbRemote24x7")
' open session
obj.OpenSession( "john_doe", "password", "WinSock", _
             "LocalHost", "1096", "", False)
' ... do something here ...
' close session destroy COM object
Set obj = Nothing
```

3. ASP example using JavaScript COM interface

```
// create COM object
var obj = Server.CreateObject("w24x7ASP.jsRemote24x7");
// open session
obj.OpenSession( "john_doe", "password", "WinSock",
             "LocalHost", "1096", "", False);
// ... do something here ...
// close session and destroy COM object
delete obj
```

ProtectJob

Long ProtectJob (String JobID,
String ProtectionType,
String JobPassword)

The **ProtectJob** function can be used to change protection type for an already protected job or setup protection on an unprotected job. If the job is already protected you must specify valid JobPassword that matches existing job password.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name
ProtectionType	Job protection code. The following values are supported: **F** – Full protection – job definition cannot be seen, changed or deleted and the job cannot be run manually. **R** – Read only protection – job definition can be seen and the job can be run manually, but the job cannot be changed or deleted. **E** – Execute only – job definition cannot be seen, changed or deleted, but the job can be run manually.
JobPassword	Job password

See also:

UnprotectJob
DisableJob
CreateJob

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.ProtectJob( "34", "R", "some password" )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.ProtectJob( "34", "R", "some password" )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.ProtectJob( "34", "R", "some password" );
if (RetCode != 1) Response.Write(obj.LastError); // Display error
```

ReleaseJob

Long ReleaseJob (Long RunJobID)

The **ReleaseJob** function releases the specified job and allows the queue to run this job. The specified job must be already waiting in the job queue and have On Hold state. Job run-time IDs are returned by GetMonitor and by GetJobQueueMonitor functions.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

RunJobID	Job Run-time ID.

See also:

HoldJob
GetMonitor
GetJobQueueMonitor

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.ReleaseJob( 15556 )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj. ReleaseJob( 15556 )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj. ReleaseJob( 15556 );
if (RetCode != 1) Response.Write(obj.LastError); // Display error
```

QueueJob

Long QueueJob (String JobID)

The **RunJob** function submits the specified job to the associated job queue. If the queue is free, the specified job starts running immediately; otherwise it waits for the queue to become free. As compared to **RunJob** function with synchronous parameter set, the submitter job does NOT wait for the submitted child job to complete.

Return: Returns unique run-time job id for the submitted job. This run-time id can be referenced in **KillJob**, **HoldJob** and other functions dealing with run-time job instances. Returns -1 if an error occurres. In case of an error use the **LastError** property to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name.

See also:

CreateJob
RunJob
RunShellCommand

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.QueueJob( "34" )
If RetCode = -1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj. QueueJob ( "34" )
If RetCode = -1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.QueueJob( "34" );
if (RetCode == -1) Response.Write(obj.LastError); // Display error
```

RunJob

Long RunJob (String JobID,
Boolean Detached)

The **RunJob** function starts the specified job. The job is executed on the computer where the target 24x7 Scheduler is running.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name.
Detached	Controls how the job is processed. **Value** **Meaning** True Start the job using a separate instance of 24x7 Scheduler and then immediately return control back to the caller False Start the job using the main instance of 24x7 Scheduler and wait for the job to complete before returning control back to the caller. **Note:** Each method has some advantages and disadvantages. If you want to wait for a job to complete before starting next job use **False** for the **Detached** property. This will ensure that jobs will run sequentially. Otherwise, if you run the job **Detached** (Detached = True) and then run another job, both of these jobs will run concurrently. On the other hand if you run a job non-**Detached** (Detached=False) and that job takes a long time to run your session may expire before the job completes and this may cause the job to abort prematurely. Always use **True** for the **Detached** property for jobs taking a long time to run.

See also:

CreateJob
QueueJob
RunScript
RunShellCommand

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.RunJob( "34", True )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.RunJob( "34", True )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.RunJob( "34", True );
if (RetCode != 1) Response.Write(obj.LastError); // Display error
```

RunScript

Long RunScript (String Script)

The **RunScript** function executes the specified JAL script. The effect of this function is similar to running a job with the same script. The script is executed on the computer running 24x7 Scheduler to which the API client session is connected to..

Important Notes: This method is only valid when executed in a client session connected to 24x7 Scheduler Windows Edition. The method is ignored if executing in a client session connected to 24x7 Scheduler Multi-platform Edition.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

Script	JAL script that you want to execute.

See also:

RunJob
RunShellCommand
UtilRunScript

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, Script As String
' Run dynamic script to copy .LOG files on the server
Script = "Dim( count, number )" & vbCrLf &_
         "FileCopyEx( ""c:\\pathA\\*.log"", ""c:\\pathB"", count )"
RetCode = obj.RunScript( Script )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, Script
' Run dynamic script to copy .LOG files on the server
Script = "Dim( count, number )" & vbCrLf &_
         "FileCopyEx( ""c:\\pathA\\*.log"", ""c:\\pathB"", count )"
RetCode = obj.RunScript( Script )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
// Run dynamic script to copy .LOG files on the server
var Script =
    "Dim( count, number )\r\n" +
```

```
    "FileCopyEx( \"c:\\\\pathA\\\\*.log\", \"c:\\\\pathB\", count )";
var RetCode = obj.RunScript( Script );
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

RunShellCommand

Long RunShellCommand (String CommandLine,
String Dir,
Boolean Async,
Long Timeout)

The **RunShellCommand** function executes the specified Operation System command or program. The command is executed on the computer where the target 24x7 Scheduler is running.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

CommandLine	The full or partial path and filename of the module to execute. The module may be a .COM, .BAT, .EXE, or associated file type. If a partial name is specified, the current drive and current directory are used by default. The module name must be the first white space-delimited token in the Program name field. The specified module can be a Win32-based application, or it can be some other type of module (for example, MS-DOS or OS/2) if the appropriate subsystem is available on the local computer. If the filename does not contain an extension, .EXE is assumed. If the filename ends in a "." with no extension, or the filename contains a path, .EXE is not appended. If the filename does not contain a directory path, Windows searches for the executable file in the following sequence: 1 The directory from which the 24x7 Scheduler loaded. 2 The current directory. 3 The 32-bit Windows system directory. The name of this directory is SYSTEM32. 4 The 16-bit Windows system directory. The name of this directory is SYSTEM. 5 The Windows directory. The name of this directory is WINDOWS or WINNT. 6 The directories that are listed in the PATH environment variable.  **Notes:** • You should always include the full path - don't rely on the PATH environment variable, because this may be different at the time the program runs, depending on what account the program is being run under. Also, keep in minds that other programs can modify the PATH environment variable as well. • Program name can be followed by command line parameters. You can use 24x7 supported macro-parameters inside program name and command line parameters String to pass dynamic information (such as the current month) to the scheduled program.
Dir	The directory where the program executable finds associated files that are required to run the program. Most programs do not require an entry in this field so that you can specify an empty string (""). Occasionally

	however, a program will refuse to run properly unless it is specifically told where to find other program files. The 24x7 Scheduler will change to this directory when running the program or document.
Async	This controls how the 24x7 Scheduler executes the process specified in the **CommandLine**. **Value** — **Meaning** True — Start the specified command and return control back to the 24x7 Scheduler. The **Timeout** argument is ignored in this case. False — Start the specified command and wait for that command to complete before returning control back to the 24x7 Scheduler. **Note:** When running the command synchronously (**Async=False**) 24x7 Scheduler enters an efficient wait state until this command finishes or the **Timeout** interval elapses. In the latter case, the 24x7 Scheduler forcedly terminates the process created by the specified **CommandLine**.
Timeout	The maximum time interval (in seconds) within which you allow the process (as specified in the **CommandLine**) to run. Use 0 **Timeout** to allow infinite waiting. The **Timeout** parameter is ignored when **Async** parameter is set to **True**.

See also:

RunJob
RunScript
UtilRunScript

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.RunShellCommand( "h:\sales\reports\month_end.exe", _
                               "", False, 30 )
If RetCode <> 1 Then
       MsgBox obj.LastError ' Display error message
Else
       MsgBox "Month-end sales reports are now ready."
End If
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.RunShellCommand( "h:\sales\reports\month_end.exe", _
                                "", False, 30 )
```

```
If RetCode <> 1 Then
       Response.Write(obj.LastError) ' Display error
Else
       ' Reports ran successfully, now redirect to the monthly
       ' report menu
       Response.Redirect("../reports/monthly/menu.asp")
End If
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.RunShellCommand(
                              "h:\\sales\\reports\\month_end.exe",
                              "", False, 30 );
if (RetCode != 1)
       Response.Write(obj.LastError); // Display error
else
       // Reports ran successfully, now redirect to the monthly
       // report menu
       Response.Redirect("../reports/monthly/menu.asp");
```

SetFolderProperty

Long SetFolderProperty (String FolderID,
String PropertyName,
String NewValue)

The **SetFolderProperty** function changes value of the specified folder **PropertyName** for the specified **FolderID**. The **PropertyName** must be a valid JDL job property name. The following folder properties are currently supported: "FOLDER_NAME", "DESCRIPTION".

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

FolderID	Folder ID converted to string or folder name.
PropertyName	The folder property name.
NewValue	Then new property value.

See also:

CreateFolder
GetFolderProperty

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
' Change description of folder "Rep12"
RetCode = obj.SetFolderPropery( "Rep12", "DESCRIPTION", _
             "This folder is now used for ad-hoc reports" )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
' Change description of folder "Rep12"
RetCode = obj.SetFolderPropery( "Rep12", "DESCRIPTION", _
             "This folder is now used for ad-hoc reports" )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...

// Change description of folder "Rep12"
var RetCode = obj.SetFolderPropery( "Rep12", "DESCRIPTION",
```

```
                "This folder is now used for ad-hoc reports" );
if (RetCode != 1) Response.Write(obj.LastError); // Display error
```

SetGlobalVariable

Long SetGlobalVariable (String VariableName,
String NewValue)

The S**etGlobalVariable** function changes the value of the global variable on the target 24x7 Remote Agent or 24x7 Master Scheduler.

 **Important Note:** This method is only valid for 24x7 Scheduler Windows Edition.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

VariableName	The name of the global variable that must exist on the target 24x7 Agent or Master Scheduler. Variable names are case-insensitive.
NewValue	The new value converted to string format.

See also:

GetGlobalVariable
RunScript

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.SetGlobalVariable( "my_variable", "125" )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.SetGlobalVariable( "my_variable", "125" )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.SetGlobalVariable( "my_variable", "125" );
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

SetJobProperty

Long SetJobProperty (String JobID,
String PropertyName,
String NewValue)

The **SetJobProperty** function changes value of the specified job **PropertyName** for the specified **JobID**. The **PropertyName** must be a valid JDL job property name.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name.
PropertyName	The job property name. For supported job properties and their JDL names see 24x7 User's Guide. As a quick reference see Job Properties in JDL Format topic in this manual.
NewValue	Then new value converted to string format.

See also:

GetJobProperty
SetJobPropertyEx
UpdateJob
SetFolderProperty
SetJobTemplateData

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
' Change value of QUEUE property for job #235
RetCode = obj.SetJobProperty( "235", "QUEUE", "Reports" )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, JobQueue
' Change value of QUEUE property for job #235
RetCode = obj.SetJobProperty( "235", "QUEUE", "Reports" )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
// Change value of QUEUE property for job #235
RetCode = obj.SetJobProperty( "235", "QUEUE", "Reports" );
if (RetCode != 1) Response.Write(obj.LastError);  // Display error
```

SetJobPropertyEx

Long SetJobProperty (String JobID,
String PropertyNameList,
String NewValueList)

The **SetJobPropertyEx** function changes values of job properties specified in the **PropertyNameList** parameter for the job specified by **JobID** parameter. The **PropertyNameList** must contain comma-separated list of valid JDL job property names.

 **Note: SetJobPropertyEx** function is an extended version of **SetJobProperty** function. **SetJobPropertyEx** is capable of updating multiple properties at once while **SetJobProperty** can update only one property at a time. **SetJobPropertyEx** provides more efficient and faster method to update multiple properties as it makes just one round-trip to the 24x7 Scheduler server. On the other hand, because new values are passed as a comma-separated list **SetJobPropertyEx** cannot be used to update values that contain commas

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name.
PropertyNameList	Comma-separated list of job property names. For supported job properties and their JDL names see 24x7 User's Guide. As a quick reference see Job Properties in JDL Format topic in this manual.
NewValueList	Tab-separated list of new property values converted to string format.

See also:

SetJobProperty
GetJobPropertyEx
UpdateJob
SetFolderProperty
SetJobTemplateData

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
      ' ... create COM object (obj) and open session here ...
      ' ... for details see OpenSession topic ...
      Dim RetCode As Long
      ' Change job #235 schedule to daily job to be run on Mondays
      ' and Wednesdays (SCHEDULE_TYPE=D, MONDAY=Y, WEDNESDAY=Y, all other
      ' days set to N)
      RetCode = obj.SetJobPropertyEx( "235", _
"SCHEDULE_TYPE,SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY", _
"D,N,Y,N,Y,N,N,N" )
```

```
      If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
      ' ... create COM object (obj) and open session here ...
      ' ... for details see OpenSession topic ...
      Dim RetCode, JobQueue
      ' Change job #235 schedule to daily job to be run on Mondays
      ' and Wednesdays (SCHEDULE_TYPE=D, MONDAY=Y, WEDNESDAY=Y, all other
      ' days set to N)
      RetCode = obj.SetJobPropertyEx( "235", _
"SCHEDULE_TYPE,SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY", _
"D,N,Y,N,Y,N,N,N" )
      If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
      // ... create COM object (obj) and open session here ...
      // ... for details see OpenSession topic ...

      // Change job #235 schedule to daily job to be run on Mondays
      // and Wednesdays (SCHEDULE_TYPE=D, MONDAY=Y, WEDNESDAY=Y, all other
      // days set to N)
      var RetCode = obj.SetJobPropertyEx( "235",
"SCHEDULE_TYPE,SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY",
"D,N,Y,N,Y,N,N,N" );
      if (RetCode != 1) Response.Write(obj.LastError);  // Display error
```

SetJobTemplateData

Long SetJobTemplateData (String JobID,
String TemplateName,
String TemplateFile,
String Data)

The **SetJobTemplateData** function saves template properties and data used to create or modify the specified job. This data can be later retrieved using the **GetJobTemplateData** function. If the function succeeds the values of **TemplateName, TemplateFile**, and **Data** parameters are saved in a file. The **Data** value is a free text that could be of virtually any size. It is up to the developer to decide what format to use for the **Data**.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name.
TemplateName	Name of the template used for the specified job.
TemplateFile	Name of the template file used for the specified job.
Data	Data that you want to save for future references.

See also:

GetJobTemplateData
SetJobProperty
SetTemplate

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, TemplateName AsString, TemplateFile As String,
    TemplateData As String

TemplateName = "Venus FTP Download"
TemplateFile = "venus_ftp.ini"
TemplateData = "SRC_DIR=\daily\cars" & vbCrLf & _
               "SRC_FILE=122003car.zip" & vbCrLf &
               "DEST_DIR=f:\carinfo"
RetCode = obj.SetJobTemplateData( "235", TemplateName, _
             TemplateFile, TemplateData  )
If RetCode <> 1  Then
       ' Display error message
       MsgBox "SetJobTemplateData error: " & obj.LastError
End If
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, TemplateName, TemplateFile, TemplateData

TemplateName = "Venus FTP Download"
TemplateFile = "venus_ftp.ini"
TemplateData = "SRC_DIR=\daily\cars" & vbCrLf & _
               "SRC_FILE=122003car.zip" & vbCrLf &
               "DEST_DIR=f:\carinfo"
RetCode = obj.SetJobTemplateData( "235", TemplateName, _
            TemplateFile, TemplateData  )
If RetCode <> 1  Then
       ' Display error message
       Response.Write("SetJobTemplateData error: " & obj.LastError)
End If
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...

var TemplateName = 'Venus FTP Download';
var TemplateFile = 'venus_ftp.ini';'
var TemplateData =  'SRC_DIR=\\daily\\cars\r\n' +
                    'SRC_FILE=122003car.zip\r\n' +
                    'DEST_DIR=f:\\carinfo';
var RetCode = obj.SetJobTemplateData( '235', TemplateName,
            TemplateFile, TemplateData  );
if (RetCode != 1)
       // display error message
       Response.Write( 'SetJobTemplateData error: ' + obj.LastError );
```

SetTemplate

Long SetTemplate (String TemplateFile,
String Buffer)

The **SetTemplate** function updates contents of the specified template file.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

TemplateFile	Template file name.
Buffer	A String variable whose value replaces contents of the specified template file.

See also:

AddTemplate
GetTemplate
GetJobTemplateData
GetTemplateCatalog
SetJobTemplateData

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
Dim TemplateCode AS String

' Copy template code from the text editor control
TemplateCode = TemplateForm.Editor.Text
' Update ftp_reports.ini template
RetCode = obj.SetTemplate( "%HOME%\Templates\ftp_reports.ini", _
                           TemplateCode )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
Dim TemplateCode

' Obtain template code from the submitted form
TemplateCode = Request.Form("TEMPLATE_CODE")
' Update ftp_reports.ini template
RetCode = obj.SetTemplate( "%HOME%\Templates\ftp_reports.ini", _
                           TemplateCode )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...

// Obtain template code from the submitted form
var TemplateCode = Request.Form("TEMPLATE_CODE");
// Update ftp_reports.ini template
var RetCode = obj.SetTemplate( "%HOME%\\Templates\\ftp_reports.ini",
                               TemplateCode );
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

Test

Long Test (Long x)

The **Test** function is provided exclusively for your convenience so you can test whether you can load and call **24x7 Remote Control COM** functions. The **Test** function does not communicate with the 24x7 servers and performs simple computations on the client side only.

Return: The returned value must be the same as the **x** value specified for the function argument.

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' Create COM object
Dim obj As Object
Set obj = CreateObject("24x7 Remote Control")

' Test it
If obj.Test(5) <> 5 Then MsgBox "Error" ' Display error message
```

2. ASP example using VBScript COM interface

```
' Create COM object
Dim obj
Set obj = Server.CreateObject("w24x7ASP.vbRemote24x7")

' Test it
If obj.Test(5) <> 5 Then _
       Response.Write("Error") ' Display error
```

3. ASP example using JavaScript COM interface

```
// Create COM object
var obj = Server.CreateObject("w24x7ASP.jsRemote24x7");

// Test it
if (obj.Test(5) != 5) Response.Write("Error") // Display error
```

UnprotectJob

Long UnprotectJob (String JobID,
String JobPassword)

The **UnprotectJob** function can be used to remove protection from a protected job. You must specify valid JobPassword that matches the existing job password.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

JobID	Job ID converted to string or job name
JobPassword	Job password

See also:

ProtectJob
EnableJob
CreateJob

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.UnprotectJob( "34", "some password" )
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.UnprotectJob( "34", "some password" )
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = obj.UnprotectJob( "34", "some password" );
if (RetCode != 1) Response.Write(obj.LastError); // Display error
```

UpdateAgentProfile

Long UpdateAgentProfile(**String ProfileName,**
String ComMethod,
String Location,
String Port,
String Options)

The **UpdateAgentProfile** function updates properties of an existing 24x7 Remote Agent or 24x7 Master Scheduler profile.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

For description of **UpdateAgentProfile** parameters see AddAgentProfile method.

See also:

AddAgentProfile
DeleteAgentProfile
GetAgentProfile

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.UpdateAgentProfile( "Print Server", "Winsock", _
                                  "192.168.100.1", "1096", "")
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.UpdateAgentProfile( "Print Server", "Winsock", _
                                  "192.168.100.1", "1096", "")
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = 0;
RetCode = obj.UpdateAgentProfile( "Print Server", "Winsock",
                                  "192.168.100.1", "1096", "");
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

UpdateDatabaseProfile

Long UpdateDatabaseProfile (String ProfileName,
String DatabaseDriver,
String Server,
String DatabaseName,
Boolean AutoCommit,
String User,
String Password)

The **UpdateDatabaseProfile** function updates properties of an existing database profile.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

For description of **UpdateDatabaseProfile** parameters see AddDatabaseProfile method.

See also:

AddDatabaseProfile
GetDatabaseProfile
DeleteDatabaseProfile

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.UpdateDatabaseProfile( "Prod Server", _
             "MS SQL Server 7.x and later", "Neptune", _
             "dataware", True, "sa", "******")
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.UpdateDatabaseProfile( "Prod Server", _
             "MS SQL Server 7.x and later", "Neptune", _
             "dataware", True, "sa", "******")
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = 0;
RetCode = obj.UpdateDatabaseProfile( "Prod Server",
             "MS SQL Server 7.x and later", "Neptune",
             "dataware", True, "sa", "******");
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

UpdateJob

Long UpdateJob (String JobID, String JobDefinition)

The **UpdatesJob** function updates properties of an existing job.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Note: **JobDefinition** parameter may contain any combination of job properties and scripts. Only values of referenced properties are updated. Values of all other properties are preserved as is.

Parameters:

JobID	Job ID converted to string or job name
JobDefinition	The Job definition in JDL format. For a wide variety of examples see job templates available in the [24x7 install directory]\Template subdirectory. The default path is *C:\Program Files\24x7 Automation 3\Template*. For supported job properties and JDL properties see 24x7 User's Guide.

See also:

CreateJob
SetJobProperty
SetJobTemplateData
ChangeFolder
DisableJob
ProtectJob
RunJob

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, JobDefinition AS String

' Copy job definition from text editor control
JobDefinition = JobDefinition.Editor.Text
' Update job #234
RetCode = obj.UpdateJob( "234", JobDefinition )
If RetCode < 0 Then
       ' Display error message
       MsgBox obj.LastError
Else
       ' Display confirmation
       MsgBox "Job updated succesfully"
End If
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, JobDefinition

' Get job definition from the submitted form
JobDefinition = JobDefinition.Editor.Text
' Update job #234
RetCode = obj.UpdateJob( "234", JobDefinition )
If RetCode < 0 Then
       ' Display error
       Response.Write(obj.LastError)
Else
       ' Display confirmation
       Response.Write("Job updated succesfully")
End If
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = 0;
// Get job definition from the submitted form
var JobDefinition = Request.Form("JOB_DEF");
// Update job #234
RetCode = obj.UpdateJob( "234", JobDefinition );
if (RetCode < 0)
       // Display error
       Response.Write(obj.LastError);
else
       // Display confirmation
       Response.Write("Job updated succesfully");
```

UpdateJobQueue

Long UpdateJobQueue (String QueueName,
Long MaxSize)

The **UpdateJobQueue** function updates properties of an existing job queue.

Note: In 24x7 Scheduler Windows Edition, updating job queue properties does not have an immediate effect. The changes are effective only after the 24x7 Scheduler is restarted. In 24x7 Scheduler Multi-platform Edition, the changes are effective immediately.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

QueueName	Name of the Job Queue whose properties you want to modify.
MaxSize	Max queue size in Mbytes. 1 Mbytes should be sufficient in most cases.

See also:

AddJobQueue
DeleteJobQueue

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.UpdateJobQueue( "Payroll jobs", 5)
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.UpdateJobQueue( "Payroll jobs", 5)
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = 0;
RetCode = obj.UpdateJobQueue( "Payroll jobs", 5);
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

UpdateJobQueueEx

Long UpdateJobQueue (String QueueName,
Long MaxSize)

The **UpdateJobQueue** function updates properties of an existing job queue.

Note: In 24x7 Scheduler Windows Edition, updating job queue properties does not have an immediate effect. The changes are effective only after the 24x7 Scheduler is restarted. In 24x7 Scheduler Multi-platform Edition, the changes are effective immediately.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

QueueName	Name of the Job Queue whose properties you want to modify.
MaxSize	Maximum allowed disk space usage in Mbytes. 1 Mbytes should be sufficient in most cases. As of 24x7 Scheduler Multi-platform Edition v4.3, this queue parameter is reserved for future use. The parameter is used in 24x7 Scheduler Windows Edition only. Specify zero value for unlimited disk space usage.
MaxJobs	Maximum number of jobs allowed in a queue at any moment in time. The count includes both running and queued jobs, including jobs placed on hold. Specify zero value for unlimited number of jobs.
NearCapacityAlerts	Enables sending email alerts in case a queue is at or near its maximum capacity. The capacity is controlled by **MaxSize** and **MaxJobs** parameters
EmailSender	Email account (for MAPI email protocol) or email address (for SMTP email protocol) to use for sending "near capacity" and "over capacity" email alerts.
EmailPassword	Email password to use for authenticating email sender to the email server. Specify an empty string or null value if password is not required.
EmailRecipients	Comma-separated list of email recipient addresses for "near capacity" and "over capacity" email alerts.

See also:

AddJobQueueEx
DeleteJobQueue

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long
RetCode = obj.UpdateJobQueueEx( "Payroll jobs", 1, 100,
             "alerter@domain.com", "", "helpdesk@domain.com")
If RetCode <> 1 Then MsgBox obj.LastError ' Display error message
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode
RetCode = obj.UpdateJobQueueEx( "Payroll jobs", 1, 100,
             "alerter@domain.com", "", "helpdesk@domain.com")
If RetCode <> 1 Then Response.Write(obj.LastError) ' Display error
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
var RetCode = 0;
RetCode = obj.UpdateJobQueueEx( "Payroll jobs", 1, 100,
             "alerter@domain.com", "", "helpdesk@domain.com");
if (RetCode != 1) /* Display error */ Response.Write(obj.LastError);
```

UtilRunScript

Standard COM and VBScript COM interfaces:
Long UtilRunScript (String Script,
ByRef String OutputBuffer)

JavaScript COM:
Long UtilRunScript (String Script)

The **UtilRunScript** function executes the specified JAL script and returns the script output. The script is executed on the computer running 24x7 Scheduler to which the API client session is connected to.
This function internally calls the RunScript function. Before calling **runScript** function to execute the **Script**, **UtilRunScript** creates a unique temporary file on the remote system running 24x7 Scheduler, and alters text of the **Script,** adding an additional variable named **output**. This variable value is set to the name the temporary file. This is how the name of the temporary file is passed to the **Script**. In the **Script** you can then use this file to save any script output. After completion of the **runScript** function, **UtilRunScript** reads contents of the remote temporary file into the **OutputBuffer** parameter and deletes the file. That's how the output saved on the remote system is returned to the caller.

Important Notes:

- This function is only valid when executed in a client session connected to 24x7 Scheduler Windows Edition. The function is ignored if executing in a client session connected to 24x7 Scheduler Multi-platform Edition.

- In JavaScript COM interface the **OutputBuffer** parameter is not used. Use the ipStringBuffer COM object property to obtain the returned output.

Note: Do not use **UtilRunScript** if you do not need to return any output from the script. Use RunScript function directly.

Return: Returns 1 if it succeeds or a negative number if an error occurs. Use the **LastError** property to obtain the error message.

Parameters:

Script	JAL script that you want to execute.
OutputBuffer This parameter is not available in JavaScript COM interface	A String variable passed by reference. In case of successful function execution, the **Script** output is copied to the **OutputBuffer**.

See also:

RunJob
RunShellCommand
RunScript

Examples:

1. Visual Basic example using 24x7 standard COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode As Long, Script As String, Output As String
' Run dynamic script to copy .LOG files on the server
Script = "Dim( count, number )" & vbCrLf & _
         "FileCopyEx( ""c:\\pathA\\*.log"", ""c:\\pathB"", count )" _
         & vbCrLf & "FileSave( output, count )"
RetCode = obj.UtilRunScript( Script, Output )
If RetCode <> 1 Then
       MsgBox obj.LastError ' Display error message
Else
       MsgBox "Total files copied: " & Output ' Display result
End If
```

2. ASP example using VBScript COM interface

```
' ... create COM object (obj) and open session here ...
' ... for details see OpenSession topic ...
Dim RetCode, Script, Output
' Run dynamic script to copy .LOG files on the server
Script = "Dim( count, number )" & vbCrLf & _
         "FileCopyEx( ""c:\\pathA\\*.log"", ""c:\\pathB"", count )" _
         & vbCrLf & "FileSave( output, count )"
RetCode = obj.UtilRunScript( Script, Output )
If RetCode <> 1 Then
       ' Display error
       Response.Write(obj.LastError)
Else
       ' Display result
       Response.Write("Total files copied: " & Output)
End If
```

3. ASP example using JavaScript COM interface

```
// ... create COM object (obj) and open session here ...
// ... for details see OpenSession topic ...
// Run dynamic script to copy .LOG files on the server
var Script = "Dim( count, number )\r\n" +
"FileCopyEx( \"c:\\\\pathA\\\\*.log\", \"c:\\\\pathB\", count )\r\n"+
"FileSave( output, count )";
var RetCode = obj.UtilRunScript( Script );
if (RetCode != 1)
       // Display error
       Response.Write(obj.LastError);
else
       // Display results
       Response.Write("Total files copied: " + obj.ipStringBuffer);
```

Job Properties in JDL Format

All job properties are documented in the 24x7 Scheduler User's Guide. This topic can be used as a quick reference for supported job properties and their JDL names.

Job Definition Language (JDL) supports the following property names:

Property Name	Meaning
ACCOUNT	E-mail Account such as user ID, profile, or e-mail address (e-mail watch job). The actual value may differ for different e-mail interfaces. For a MAPI interface you should use the name of the MAPI profile you use when logging on to the e-mail system. For Lotus Notes you should use the name of the user (or ID) you use when logging on to the Lotus Notes. For SMTP you should use your e-mail address.
AGENT	Same as Host (see Host description)
ALL_DAY_TYPE	All Day Schedule Type, one of the following: **R**, **L** (**R** - recursive, repeat at specified intervals; **L** - fixed time list)
ARCHIVE_OUTPUT	Capture and archive job output, one of the following: **Y**, **N** (yes, no) **Note**: This property is not used in 24x7 Scheduler Windows-only Edition.
ASYNC	Asynchronous Process, one of the following: **Y**, **N** (yes, no)
BACKUP_AGENT	Same as Backup Host (see Backup Host description)
BACKUP_HOST	Backup Remote Host (e.g. Backup Remote Agent name)
CALENDAR	Name of the Calendar object assigned to the job
COMMAND	Program Command Line
DAY_END_TIME	Daily End Time for "all day" jobs with limited run-time interval
DAY_LIST	Monthly Schedule List of fixed Day Numbers, numbers must be in 1..31 range. Example: 1,3,5,7,14. This property is shared with TIME_LIST property for All Day Schedule.
DAY_NAME	Monthly Schedule Day Name, one of the following: **Monday , Tuesday, Wednesday , Thursday , Friday , Saturday , Sunday , Weekend, Weekday**
DAY_NUMBER	Monthly Schedule Day Number, a number from 1 - 31 range
DAY_START_TIME	Daily Start Time for "all day" jobs with limited run-time interval
DELAY	Allowed Job Delay Interval (minutes)
DELETE_RULE	Delete, Move, and Rename Semaphore File Rules, one of the following: D, **A, B, M, E, R, C, F, G** (**D** - do not delete, move, rename; **A** - delete after job run; **B** - delete before job run; **M** - move before job run; **E** - move after job run; **R** - rename before job run; **C** - rename after job run; **F** -move and rename before job run; **G** - move and rename after job run)
DESCRIPTION	Job Description
DISABLE_ON_ERROR	Disable Job on Error, one of the following: **Y**, **N** (yes, no)
DISABLED	Job Disabled Status, one of the following: **Y**, **N** (yes, no)
DETACHED	Detached Job, one of the following: **Y**, **N** (yes, no) **Note**: This property is not used in 24x7 Scheduler Multi-

	platform Edition.
END_DATE	Last Job Start Date
END_TIME	Last Job Start Date
EXIT_CODE	Exit Code Condition (as a string expression)
FILE	Semaphore File Names(s) for file-watch jobs; Module Name for process-watch job
FOLDER	Job Folder ID. This is read-only property. It may not be changed using **SET** command. It can be retrieved using **GET** command
FOLDER_NAME	Job Folder Name. This is read-only property. It may not be changed using **SET** command. It can be retrieved using **GET** command
FRIDAY	Execute Job On Fridays, one of the following: **Y**, **N** (yes, no)
HOST	Remote Host (Remote Agent Name)
ID	Job ID, This is read-only property. It may not be changed using **SET** command. It can be retrieved using **GET** command with Job Name parameter.
IGNORE_ERRORS	Ignore Errors, one of the following: **Y**, **N** (yes, no)
INIT_TIMEOUT	Initial Timeout before sending keystroke (seconds)
INTERVAL	Repeat Interval for Job having Schedule Type **T**
JOB_PASSWORD	Job Protection State and Password. Sets or removes job protection state and password. This is a write-only property. It can be changed using **SET** command, but it cannot be retrieved using **GET** command. The value in this property must be specified in the following format : [old password][tab character][new password][tab character][protection state] If the job is not protected, the [old password] is ignored, otherwise a valid password must be specified in order to remove or change job password or protection state. If the protection exists and the new protection state is specified as an empty string the protection will be removed. The protection code must one of the following: **F**, **E**, **R**, an empty string (**F** -full protection; **E** - execute only; **R** - read only; an empty string indicates that a job is not protected).
JOB_TYPE	Job Type, one of the following: **P**, **D**, **S** (program, database, script)
KEYSTROKE	Keystroke to send to the launched program.  **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
LOG	Log Job Execution, one of the following: **Y**, **N** (yes, no)
MESSAGE	E-mail Message Text (e-mail watch job)
MODIFY_TERMINAL	Network name of the computer from which the job was last modified. This is a read-only property. It cannot be changed using **SET** command. It can be retrieved using **GET** command.  **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
MODIFY_TIME	Date and time when the job was modified. This is a read-only property. It cannot be changed using **SET** command. It can be retrieved using **GET** command.  **Note**: This property is not used in 24x7 Scheduler Multi-

	platform Edition.
MODIFY_USER	Name of the user who last modified the job. This is a read-only property. It cannot be changed using **SET** command. It can be retrieved using **GET** command.  **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition
MONDAY	Execute Job On Mondays, one of the following: **Y**, **N** (yes, no)
MONTHLY_TYPE	Monthly Schedule Type, one of the following: **D**, **T**, **L** (**D** - by day number; **T** - by day name; **L** - fixed day list)
MOVE_DIR	Name of the destination directory for semaphore file move and rename operations.
MSG_ACCOUNT	E-mail Account for Notification Action of E-mail Type E-mail (user ID, profile, or e-mail address). The actual value may differ for different e-mail interfaces. For the MAPI interface you should use the name of the MAPI profile you use when logging on to the e-mail system. For Lotus Notes you should use the name of the user (or ID) you use when logging on to Lotus Notes. For SMTP you should use your e-mail address.
MSG_ACTIONS	Map of Notification Actions and Events in text format. The map is represented as a comma-separated list of 2-character values where in every list item the first character represents Notification Event Type and the second character represents Notification Action Type. The following characters can be used for the event type: **S** - job start, **F** - job finish, **E** - job error, **N** - job file not found, **L** - job is late. The following characters can be used for the action type: **E** - send email, **P** - send page, **N** - send network popup message, **D** - execute database commands, **F** - create semaphore files, **T** - send SNMP trap, **J** - run job, **R** - run program, **S** -run script. Example map: SE,FE,EE,FD This example map represents the following Notification Events and Events: 1. Send notification email on job start. 2. Send notification email on job finish. 3. Send notification email on job error. 4. Execute database commands on job finish.
MSG_DATABASE	Execute Notification Action of Database Type, one of the following: **Y**, **N** (yes, no)
MSG_E-MAIL	Execute Notification Action of E-mail Type, one of the following: **Y**, **N** (yes, no)
MSG_ERROR	Execute Notification Action on Job Execution Error, one of the following: **Y**, **N** (yes, no)
MSG_FILE	Execute Notification Action of Semaphore File Type, one of the following: **Y**, **N** (yes, no)
MSG_FILE_NAME	File name(s) for Notification Action of Semaphore File Type
MSG_FINISH	Execute Notification Action on Job Finish, one of the following: **Y**, **N** (yes, no)
MSG_JOB	Execute Notification Action of Run Job Type, one of the following: **Y**, **N** (yes, no)
MSG_JOB_ID	Job name or job id for Notification Action of Run Job Type

MSG_LATE	Execute Notification Action on Job Late Start, one of the following: **Y**, **N** (yes, no)
MSG_NET	Execute Notification Action of Network Message Type, one of the following: **Y**, **N** (yes, no)  **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
MSG_NET_RECIPIENT	Message Recipient for Notification Action of Network Message Type  **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
MSG_NOTFOUND	Execute Notification Action on Job Executable Not Found Error, one of the following: **Y**, **N** (yes, no)
MSG_PAGE	Execute Notification Action of Page Type, one of the following: **Y**, **N** (yes, no)  **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
MSG_PAGER	Page Recipient's Pager Number  **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
MSG_PASSWORD	E-mail Password for Notification Action of E-mail Type
MSG_PROFILE	Database Profile for Notification Action of Database Type
MSG_PROGRAM	Execute Notification Action of Run Program Type, one of the following: **Y**, **N** (yes, no)
MSG_PROGRAM_NAME	Program name for Notification Action of Run Program Type
MSG_PROGRAM_TIMEOUT	Process Timeout (seconds) for Notification Action of Run Program Type
MSG_RECIPIENT	E-mail Recipient for Notification Action of E-mail Type
MSG_SCRIPT	Script for Notification Action of Script Type  **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
MSG_SCRIPT_ TYPE	Type of Script for Notification Action of Script Type, one of the following: **JAL**, **VBS** (Job Automation Language, Visual Basic Script)  **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
MSG_START	Execute Notification Action on Job Start, one of the following: **Y**, **N** (yes, no)
MSG_TRAP	Execute Notification Action of Send SNMP Trap Type, one of the following: **Y**, **N** (yes, no) NAME Job Name
MULTI_INSTANCE_CONTROL	Job queuing rule for handling multiple job instances , one of the following: R – always queue and run, **T** – terminate already running and queued job instances if any, and add new instance, **S** – if there are already running or queued job instances, skip new instance and do nothing, **E** – if there are already running or queued job instances, skip new instance and raise an error.
NAME	Job Name

OUTPUT_PATTERN	Job Output evaluation pattern This property is not used in 24x7 Scheduler Windows-only Edition.
PASSWORD	E-mail Password (e-mail watch job)
POLLING_INTERVAL	Polling Interval (minutes)
PRIORITY	Job Priority, one of the following: **-1** – low, **0** – normal, **1** – high
PROFILE	Database Profile
PROTECTION	Job Protection State, one of the following: **F**, **E**, **R**, an empty string (**F** – full protection; **E** – execute only; **R** – read only; an empty string indicates that a job is not protected). This is a read-only property. It cannot be changed using **SET** command. It can be retrieved using **GET** command.
QUEUE	Job Queue
REBOOT	Reboot Computer After Job Finished, one of the following: **Y**, **N** (yes, no) **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
RETRY_INTERVAL	Retry Interval (seconds)
RENAME_SUFFIX	Retry Interval (seconds)
RETRY_ON_ERROR	The name suffix used in semaphore file names for rename operations.
RUNAS_DOMAIN	Domain Name for authentication of jobs to be run using another user account. **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition. In that version the RUNAS_USER must contain both the domain and user names in **domain\user** format.
RUNAS_PASSWORD	Password for authentication of jobs to be run using another user account.
RUNAS_USER	User Name for authentication of jobs to be run using another user account.
SATURDAY	Execute Job On Saturdays, one of the following: **Y**, **N** (yes, no)
SAVE_ATTACHMENT	Save E-mail Attachments (e-mail watch job), one of the following: **Y**, **N** (yes, no)
SCHEDULE_TYPE	Schedule Type, one of the following: **O**, **D**, **T**, **M**, **F**, **P**, **A**, **E**, **I**, **L**, **S** (Time trigger: **O** – run once, **D** – repeat daily, **T** – repeat at specified time interval, **M** – repeat monthly; File trigger: **F** – check semaphore files; Process trigger: **P** – check process presence, **A** – check process absence; **E**-mail trigger: **E** - check e-mail message; User trigger: **I** – wake up on "user idle" event, **L**- wake up on log-off event, **S** – wake up on shutdown event)
SCRIPT	Job Script
SCRIPT_TYPE	Job Script Type, one of the following: **JAL**, **VBS**, **JS** (Job Automation Language, Visual Basic Script, JavaScript) **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.
SEND_KEYSTROKE	Send Keystroke, one of the following: **Y**, **N** (yes, no)

	Note: This property is not used in 24x7 Scheduler Multi-platform Edition.
SIZE_CHECK_INTERVAL	File Size Stability Check Interval (used in File-watch jobs)
SKIP	Skip Late Job, one of the following: **Y**, **N** (yes, no)
SKIP_HOLIDAY	Skip Job on Holiday, one of the following: **Y**, **N** (yes, no)
SLIDE_HOLIDAY	Slide Job Execution Time on the next non-holiday if it falls on holiday, one of the following: **Y**, **N** (yes, no)
SQL	SQL Command(s)
START_DATE	First Start Date
START_IN	Program Start-up Directory
START_TIME	First Start Time
SUBJECT	E-mail Message Subject for (e-mail watch job)
SUNDAY	Execute Job On Sundays, one of the following: **Y**, **N** (yes, no)
TIME_LIST	All Day Schedule List of fixed Times, values must be in valid 24-hour time format. Example: 11:10,12:10,17:10,18:10. This property is shared with DAY_LIST property for Monthly Schedule.
THURSDAY	Execute Job On Thursdays, one of the following: **Y**, **N** (yes, no)
TIMEOUT	Timeout (seconds)
TUESDAY	Execute Job On Tuesdays, one of the following: **Y**, **N** (yes, no)
WEDNESDAY	Execute Job On Tuesdays, one of the following: **Y**, **N** (yes, no)
WINDOW	Window Style, one of the following: **N**, **M**, **I**, **H** (normal, maximized, iconic, hidden) **Note**: This property is not used in 24x7 Scheduler Multi-platform Edition.

Installation

Minimal System Requirements

1. 24x7 Scheduler Multi-platform Edition v4.0 or later
2. 350 Kbytes additional free disk space

Installation Steps

Run 24x7 Automation Suite installation program **24x7mp_setup.exe**. Follow instructions provided by the installation program.

Security Issues

Use existing Windows security features to restrict access to the 24x7 Scheduler files installed on the computer where you are invoking 24x7 Remote Control COM interface. By protecting access to these files you prevent unauthorized users from accessing and using 24x7 Remote Control COM.

To secure access to the remote 24x7 Scheduler servers enable security options on these servers. For more information please see "Security" topic in the 24x7 Automation Suite User's Guide and on-line help files.

Licensing

1. **Single installation license:** A separate single installation license is required for every 24x7 Remote Control COM installation. That's it, you need at least 2 single licenses (or a site license) for using 24x7 Remote Control COM to control and manipulate 24x7 Scheduler or 24x7 Remote Agent. In case of 2 single licenses, one license must be applied to the 24x7 Scheduler server or 24x7 Remote Agent and another license must be applied to the 24x7 Remote Control COM.
2. **Site license:** 24x7 Scheduler site license covers unlimited usage of 24x7 Remote Control COM. Site license users can use 24x7 Remote Control COM just as they use other components of the 24x7 Automation Suite. The usage of 24x7 Scheduler, 24x7 Remote Agents, 24x7 Remote Control, 24x7 Remote Control Java and 24x7 Remote Control COM is governed by their site license agreement. The site license also allows installing and using 24x7 Remote Control COM on **one** Web server. You may not install or run it on multiple servers using the same license.
3. **Web server usage:** You must obtain 24x7 Scheduler site license before you can use 24x7 Remote Control COM on your server. A separate site license is required for every web server utilizing 24x7 Remote Control COM.
4. **Redistribution:** You must obtain 24x7 Remote Control COM redistribution license before you can redistribute it with your program.

The LICENSE.TXT file congaing text of the single installation license can be found in the 24x7 Scheduler home directory. If you would like to obtain the redistribution license, please email your request to sales@softtreetech.com.