# 24x7 Scheduler™ 3.6

## User's Guide and Job Automation Language (JAL) Reference

Copyright © SoftTree Technologies, Inc. 1998-2009
All rights reserved

# Table of Contents

# About This Guide

This user's guide describes all features of the 24x7 Scheduler, an advanced job scheduling and automation system for Microsoft Windows 95/98/Me/NT/2000/XP/VISTA/2003/Vista/2008 platforms. Information in this manual applies to the 24x7 Scheduler v3.4.29 running on all supported operating systems. This manual contains information for both beginning and experienced users of the 24x7 Scheduler. Both the print and the on-line documentation assume that you have a working knowledge of standard Windows mouse and keyboard actions and understand Windows basic concepts. This manual is provided so that the reader can understand how 24x7 Scheduler functions. It also contains information on the following topics:

- Installation and configuration instructions
- Task-oriented guidelines to all interactive 24x7 Scheduler functionality
- A complete reference for the Job Automation Language including numerous samples
- Descriptions of supported programming interface methods for integrating 24x7 Scheduler with other applications and systems

The 24x7 Scheduler documentation consists of this manual and a complete 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.

User Input
Keys are rendered in **bold** to stand out from other text. Key combinations that are meant to be typed simultaneously are rendered with "+" sign between the keys, such as:
**Ctrl+F**
Keys that are meant to be typed in sequence will be separated with commas, for example:
**Alt+S, H**
This would mean that the user is expected to type the Alt and S keys simultaneously and then to type the H key.

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 FTP, HTTP, RAS, SQL, DBMS, SSH and other.

# Trademarks

24x7 Automation Suite, 24x7 Scheduler, 24x7 Event Server, DB Audit, DB Audit Expert, DB Mail for Oracle, 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 Graphics, Inc. AIX is a trademark of International Business Machines, Inc. AT&T is a trademark of American Telephone and Telegraph, Inc.
Microsoft SQL Server is a registered trademark of Microsoft Corporation.
Oracle is a registered trademark of Oracle Corporation.
IBM, DB2, UDB are registered trademarks of International Business Machines Corporation
All other trademarks appearing in this document are trademarks of their respective owners. All rights reserved.

# Chapter 1: Getting Started

## Overview

The 24x7 Scheduler is the tool that enables you to schedule tasks (jobs) to run regularly, when it is necessary for you. The 24x7 Scheduler functions are used to submit and manage jobs to be executed at a given computer at a given time or times in future or at a given event such as creation of one or more files, receiving of e-mail messages, and other process interruptions. Jobs can be managed at remote and local computers provided schedule service is running at a given computer.

Schedule service functions were designed to extend basic functionality found in many commercial programs including native Window NT character-based "AT" command. These utilities are relatively primitive when it comes to scheduling critical production jobs, because they are limited to a single machine, require cryptic instructions, and lack any error-handling, logging, or notification capabilities.

The 24x7 Scheduler enables you to:

- Schedule a job to run based on date and time, files arrival, and/or job dependencies (managed via semaphore files) such as job success, failure , missing file, etc.
- Change the schedule for or turn off an existing job.
- Customize how a job will run at its scheduled time.
- Monitor job execution progress in a real-time as well as forecast job start.
- Organize logically related jobs into logical groups represented by folders.
- Maintain list of exception dates such as holidays.
- Manage job interdependencies.
- Send notification messages about job execution status.
- Log job execution progress and status.
- Generate detailed reports on the job definitions, dependencies, and execution status.
- Use comprehensive fault tolerance features that ensure an automatic server rollover in the event of a network/machine failure.
- Integrate the 24x7 Scheduler with third-party applications.

The 24x7 Scheduler allows you to schedule a job to run at certain times. You can specify a condition that will trigger the job start. The following trigger types can be used:

- Time watch
- File watch
- Process watch
- E-mail watch
- Log-off/shutdown watch
- User-defined triggers

To start the 24x7 Scheduler each time you start Windows and run it in the background simply add the 24x7 Scheduler shortcut to the StartUp system folder.

On Windows NT 4, 2000, XP, Vista, 2003, 2008, 7 platforms 24x7 Scheduler provides an option to install 24x7 Windows NT service. This allows you to start 24x7 automatically when Windows NT (NT/2000/XP/VISTA/2003/2008) starts regardless of user logon. If setup under system account the 24x7 service will run even when the user is logged off.

When the 24x7 Scheduler is running minimized (as a scheduling service) this  icon appears on the taskbar. You can double-click the 24x7 Scheduler icon on the taskbar to open the 24x7 Scheduler. You can also use right click on the 24x7 Scheduler icon to invoke context menu from which you choose **Restore** command.

**Running on network**

The 24x7 Scheduler is designed to run effectively in a network environment. However, it is not recommended to run shared 24x7 executables. The program automatically finds its home directory where it stores schedule database, creates log files and other working files.

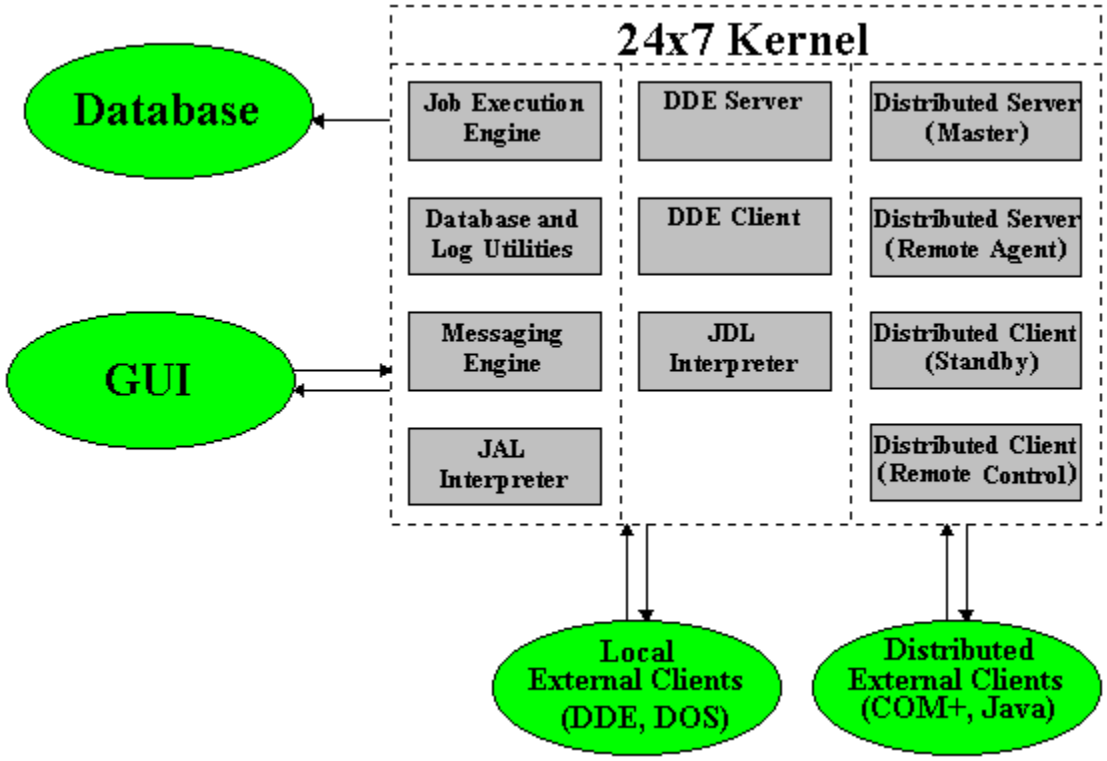
**Security issues**

If you are running the 24x7 Scheduler on a network server, on a computer where someone other than the system administrator may have access to, you should consider possible security risks when scheduling a program. If the 24x7 Scheduler is running on a computer only accessible to the system administrator, security is not a concern. Otherwise, consider the following security steps:

- Whenever possible, run any scheduled programs non-interactively (choose **hidden** window option).
- If a program must be run interactively, run it under a user account that has only the minimum authority needed to run the program properly.

**24x7 Scheduler Architecture**

You can see the 24x7 Scheduler logical architecture on the diagram below.




**See also:**

Job Explorer
Job Properties Wizard

## To Start 24x7 Scheduler Each Time Windows Starts

1 Click the **Start** button, and then point to the **Settings**.
2 Click **Taskbar**, and then click the **Start Menu Programs** tab.
3 Click **Add**, and then click **Browse**.
4 Locate **24x7.EXE**, then double-click it.
5 Click **Next**, and then double-click the **StartUp** folder.
6 Type the name (such as “24x7 Scheduler" ) that you want to see on the **StartUp** menu, and then click **Finish**.

## Running 24x7 Scheduler as a Windows NT (NT/2000/XP/2003/VISTA/2008) Service

The 24x7 Scheduler can be optionally set to run as a Windows NT (NT/2000/XP/2003/VISTA/2008/7) service. There are several important Windows NT service features that you should know and carefully consider before setting the 24x7 Scheduler to run as a service:

- The 24x7 Scheduler can start automatically whenever the computer is started and runs continuously in the background, regardless of whether a user is logged on.

- Programs run by the 24x7 schedule service inherit the security attributes of the scheduler. If the 24x7 schedule service is set to log on using local system account, the 24x7 schedule service acts as if it is part of the operating system. Therefore, all started programs get the same security permissions as the operating system. The 24x7 schedule service can be also set to using any valid Windows NT user account, so started programs will inherit the security profile of that user.
- Not all processes, applications, documents, and .bat files can be started from the 24x7 Scheduler. In order to run an interactive application, the application needs to be started with the local system account, and the "Allow service to interact with desktop" check box in Services Control Panel Applet must be checked. If the 24x7 schedule service is set to log on as the local system, scheduled jobs cannot be validated on the network so they cannot access any network resources. If you start the 24x7 schedule service with a user account, jobs can be validated on the network, but they can't have any user interface, because only the local system has sufficient privileges to allow a service to start a program on the interactive desktop. Unless 16-bit applications have access to the interactive desktop, they most likely not start as they run under Ntvdm. Therefore, you can have network access or interactivity with the system (needed by Ntvdm), but not both.

  **All these limitations are by Windows NT (NT/2000/XP/2003/VISTA/2008/7) design, for security purposes. You do not want a regular user to be able to schedule a job that would run when the administrator is logged on, and use the administrator's credentials.**
- When the 24x7 Scheduler runs as a service, you cannot do much with it other than stop it and start it. Instead, you use the standalone version of the 24x7 Scheduler to modify job database and customize 24x7 Scheduler options.

**Tip:** The 24x7 schedule service is not installed and configured automatically on installation. For information on installing and configuring the 24x7 Scheduler to run as a Windows NT service, see "Options' and "Installation and Uninstallation" help topics.

For information on Windows NT (NT/2000/XP/2003/VISTA/2008/7) services, see your Windows NT documentation. You may also want to visit Microsoft technical support on the Web. There you can find lots of information about Windows NT services. We recommend you to check the following Microsoft knowledge base articles:

Q124184 - Service Running as System Account Fails Accessing Network.
Q132679 - Local System Account and Null Sessions in Windows NT.
Q152451 - Applications Run from the Schedule Service Fail to Print.
Q158825 - System and User Account Difference with AT Command.

**See also:**

System Requirements
Installation and Uninstallation
Service Options

# Running 24x7 Scheduler as a Windows 95/98/Me Service

The 24x7 Scheduler can be optionally set to run as a Windows 95/98/Me service application. There are several important features that you should know and carefully consider before setting the 24x7 Scheduler to run as a service:

- The 24x7 Scheduler can start automatically whenever the computer is started and run regardless of whether a user is logged on.
- A user can stop the 24x7 Scheduler service by right-clicking on the 24x7 Scheduler icon in the system tray, then selecting **Exit** command from the popup menu.
- When the 24x7 Scheduler runs as a service with "survive log off" option enabled, it can run other applications in the event of a user log-off. This can be done using jobs scheduled "on user log-off" . Such started applications run regardless of whether a user is logged on.

**Tip:** The 24x7 schedule service is not installed and configured automatically on installation. For information on installing and configuring the 24x7 Scheduler to run as a Windows 95/98/Me service application, see "Options" and "Installation and Uninstallation' help topics.

**See also:**

System Requirements
Installation and Uninstallation
Service Options

# Job Explorer

The 24x7 Job Explorer presents jobs as a hierarchical structure "Tree View" on the left side. This side is filled with folders and jobs. The right side displays the properties of a selected folder or job. This side is blank unless there is at least one job in the selected folder. You can use tabs on the top of right side to change **Properties** view.

Navigating through the various folders and jobs is usually accomplished by clicking individual folders and jobs with the mouse. For users more accustomed to keyboard navigation, The 24x7 Scheduler has been designed to make every feature available via the keyboard.

To change the size of either side of the window, drag the bar that separates the two sides. Use scroll bars to navigate both sides of the Job Explorer window.

You can click your right mouse button anywhere to see a menu of available commands. The appeared context menu shows the most frequently used commands for that job, folder, or tab.

**Explorer "Tree View"**

If a folder has been expanded, and its contents displayed in the **Properties** view area, the folder will be represented by an open folder icon .

Collapsed folders represented by a closed folder icon .

Folders with a "+"symbol next to the folder name mean that there are jobs beneath the folder.

Conversely, a "-"symbol next to a folder icon means that there are no further jobs beneath.

**Property Pages**

The first tab **Jobs** displays large icons representing all jobs from a selected folder, so that you can easily select and double-click them. Double-click the job icon to launch the Job Wizard, which will help you to change job's properties. Alternatively, you can press F4.

Click a folder on the left side of the window to display its contents on the **Jobs** tab. Click the plus signs (+) to expand a folder or alternatively, you can double-click the folder or press F5 key. Click the minus signs (-) to collapse a folder alternatively, you can double-click the folder or press F6 key. To expand / collapse all folders select Expand All / Collapse All commands from the programs View menu.

The second tab **Properties** displays detailed information about the selected job. To select a job, click on the jobs icon in the left pane or switch to the **Jobs** tab then select a job there. The **Properties** tab is disabled when there is no job selected. For example, when the current folder is empty.

The third tab **Log** displays available log records for the selected job. The job log provides a complete audit trail for all job runs. The **Log** tab is disabled when there is no job selected. For example, when the current folder is empty.

**Status Bar**

The status bar is a horizontal area that appears on the bottom of Job Explorer window.

The status bar provides information about the current state of what you are viewing in the window and any other contextual information.


The status bar is divided into four sections. Most left section displays contextual information and descriptions of the program's activities during its various operations.

The second section displays small resource meter that reflects free system resources. The height of the green bar is proportional to amount of free system resource. The bar becomes yellow when there is less then 15% of free resources, and eventually it becomes red when there is less then 10% resources available. You can double-click the bar or rest mouse pointer over the bar for a moment to see additional information.

The third section displays countdown for the next pending job. If there is no job pending this section shows "Off". You can rest the mouse pointer over this section for a moment to view additional information about this job. Double-click on this section to locate and highlight the first pending job.

The last section displays the current date and time. You can double-click this section bar or rest the mouse pointer over it for a moment to view additional information.

**System Tray Icon**

The system tray is the small panel in the lower right corner of the Windows Taskbar. Job Explorer places an icon in the system tray when it is running in the background. Job Explorer is always hidden when minimized. You can right-click on the system tray icon and a context menu will be displayed. This context menu has three menu items:

1 **Restore** - brings up the Job Explorer window
2 **Close** - closes the Job Explorer window and exits Scheduler
3 **About** - displays information about 24x7 Scheduler

**Tips:**

- Use CTRL TAB shortcut to switch between tabs on the right side of the Job Explorer window.
- For brief view of job descriptions, rest the mouse pointer over the job icon on the left side of the Job Explorer window. A description of that job will appear as a ToolTip next to the mouse pointer.
- Double-click on the 24x7 system tray icon to restore the Job Explorer window.

**See also:**

Drag and Drop Interface
Support for Windows Explorer Drag and Drop
Adding New Job
Deleting Job
Disabling/Enabling Job
Modifying Job Definition and Schedule
Moving Job to Another Folder
Testing Job Execution

# Job Types

There are three types of jobs:

- Program Files and Documents
- Database Commands
- Job Automation Scripts

**Program Files and Documents**

This job type includes jobs that execute operation system files. Files than can be executed are program files such as .exe, .bat, and .com files or document files such as MS Access database systems, Perl scripts, Java scripts, etc.

Note that for jobs which execute a "document" file e.g. x:\feed\replica.mdb, the 24x7 Scheduler automatically determines which "driver" application will be used to start the process.

**Database Commands**

This type includes jobs that can execute one or more database commands. After establishing a database connection, the 24x7 Scheduler sends the specified SQL in one complete pass. Before using a multi-statement SQL, make sure that your database is capable of processing it.

You must create at least one database profile before you can run jobs of this type.

**Job Automation Scripts**

This type includes jobs that can execute user-defined scripts. The 24x7 Scheduler supports two scripting languages: Job Automation Language (i.e. JAL) and Visual Basic Script (i.e. VBS). The 24x7 Scheduler parses the script, checks syntax errors, then interprets and executes the script line by line. See Job Automation Language Help for details on JAL. See Visual Basic Help for details on VBS.

The 24x7 Scheduler has a very sophisticated built-in script Editor, which you can use to develop both SQL, JAL and VBS scripts.

# File Types

To create a file type:

1 Double-click the **My Computer** icon or start Windows Explorer.
2 On the **View** menu, click **Options**, and then **File Types** tab.
3 To create a new file type, click **New Type**.
4 Specify a description for the file type and the filename extension associated with this type of file.
5 Click **New** to define an action for this file type.

To modify the settings belonging to an existing file type:

1 Double-click the **My Computer** icon or start Windows Explorer.
2 On the **View** menu, click **Options**, and then **File Types** tab.
3 Select the desired type, then click **Edit.**
4 Click the command in the **Action** box that you want to modify, then click **Edit**.
5 Specify the action that you want to define, such as **Open** or **Print**, and the command that should run in order to complete this action.
6 Repeat steps 4 and 5 for as many actions as you want to define for this file type.

If you want to change which program starts when you open a file, carry out the following steps:

1 In **My Computer** or Windows Explorer, click on the **View** menu, then click **Options**.
2 Click the **File Types** tab.
3 In the list of file types, click the one you want to change.

The settings for that file type are shown in the **File Type Details** box.

4 Click **Edit**.

5 In the **Actions** box, click **Open**.
6 Click **Edit**, and then specify the program you want to use to open files that have this extension. This program is the "driver" application that the 24x7 Scheduler uses to run the scheduled document.

**Tip:**
- The 24x7 Scheduler uses a "driver" application default icon to represent a document in the Job Explorer.

# Drag and Drop Interface

24x7 Job Explorer supports a standard drag and drop interface for managing logical job organization. When you drag a job icon in the Job Explorer tree and drop this job into another folder, the 24x7 Scheduler moves this job to the targeted folder. When you drag a folder icon and drop it in another folder, the 24x7 Scheduler moves all jobs from this folder to the targeted folder then removes the dragged folder from the database.

If you want to move a job to another folder:

1 Click on the desired job.
2 While holding down the left mouse button, drag (move) the mouse pointer to the desired folder.
3 Release the mouse button to drop the job.

When moving all jobs from one folder to another folder:

1 Click on the desired folder.
2 While holding down the left mouse button, drag (move) the mouse pointer to the desired folder.
3 Release the mouse button to drop the folder.

**See also:**
Job Explorer
Support for Windows Explorer Drag and Drop

# Support for Windows Explorer Drag and Drop

The 24x7 Scheduler can interface with Windows Explorer by using the drag and drop interface to add new events to the loaded schedule database. When you drag a valid program or document file and drop it into Job Explorer, 24x7 Scheduler will add this file as a new event to the targeted folder. When you drag a valid file folder into Job Explorer, the 24x7 Scheduler will add this folder as a new job folder. The 24x7 Scheduler can process only one file at a time.

You can add a file to the loaded schedule database by following these steps:
1 Start Windows Explorer or File Manager
2 While holding down the left mouse button, drag (move) the mouse pointer to the Job Explorer.
3 Release the mouse button to drop the file.

**See also:**
Job Explorer
Support for Windows Explorer Drag and Drop

# Semaphore Files

A semaphore file is a type of synchronization object that can be used to effectively control the flow of processes. A data file or a log file created by one process can serve at the same time as a semaphore files for other processes. The 24x7 Scheduler can use such semaphores as a reliable method of processing job interdependencies. No matter what occurs in the previous job, the next job in line will not start until all the necessary semaphore files have been created. The semaphore files mechanism guarantees that whether you restart the 24x7 Scheduler, reboot the computer running the 24x7 Scheduler, or a network failure occurred, a dependent job will start only when all the specified files exist.

There are two kinds of the semaphore files:

- **Input files**
- **Output files**

An **input file** is a “watch-file". 24x7 Scheduler checks whether this is present in order that the dependent job can start. The 24x7 Scheduler is capable of checking the presence of one more files for the every scheduled job that relies on them. When you need to specify that more than one file should be checked, make sure that you use a list of semaphore files separated by commas.

When scheduling a job, you can use the **Polling Interval** property to specify how frequently you want 24x7 Scheduler to check the input semaphore files. Carefully consider the value for the polling interval. A short polling interval allows early detection of new semaphore files, thus the dependent jobs can start almost immediately after semaphore files arrived. On the other hand, a short interval causes the 24x7 Scheduler to check for the semaphore files more often which leads to more network traffic and takes more CPU time, leaving less CPU time to other processes.

An **output file** is a file created by the 24x7 Scheduler when a specified event is detected. You should use a list of semaphore files separated by commas when you need to specify that more than one file is to be created.

**See also:**

Job Execution Properties
About Job Interdependencies
Dependencies Editor Interface

# Execution Logs

**About main job execution log**

The 24x7 Scheduler automatically performs event logging. By default, it creates entries in the main log file only. The main log file provides a complete audit trail for all job runs. Log entries are created for the following job activities: job start, job finish, and job error. In addition, the 24x7 Scheduler logs all unexpected errors. The main log shows the date and time a job was active, the job number and name, the event severity, and the event description. The event description may include error messages produced when an error occurred.

Use the Log Viewer to view all available entries in the log file. From time to time you should clean up the main log file so that it does not grow too large (see Deleting Log in the Log Viewer topic). Choose **View / Log** command from the menu to start Log Viewer (keyboard shortcut CTRL L).

**About NT (NT/2000/XP/2003/VISTA/2008/7) system event log**

The 24x7 Scheduler can optionally log all job entries in the Windows NT Application Log. This feature is only available for Window NT/2000/XP/2003/VISTA/2008/7 platforms. For detail on the Windows NT Application Log, click the **Start** button, then choose the **Help** command from the Windows NT Start menu. You can turn on/off logging to the NT log file in the program **Options**.

To turn on/off logging in Windows NT (NT/2000/XP/2003/VISTA/2008/7) Application Log:

1 Select the **Tools** command then select **Options**.
2 Click on the **Log** tab.

3 Check or uncheck **Logging to the system event log enabled** option.

**About Status Report**

The 24x7 Scheduler is capable of duplicating main log entries in HTML report format and automatically updating this report each time a new entry is added to the log file. This report can be placed on your company Web server, where you can view it using any Web browser that supports frames. This will enable you to monitor job execution over the Internet. Alternatively, you can launch the default Web browser from the menu using **View/Status Report** command.

To enable/disable Status Report updating:
1 Select the **Tools** command then select **Options**.
2 Check or uncheck **Generate Status Report (HTML)** option on the **General** tab.

**About other log files**

When tracing is turned on, the 24x7 Scheduler creates other log files in which it stores various trace information. These log files can be viewed in the Log Viewer mentioned above.

To turn on/off tracing:
1 Select the **Tools** command then click on **Options**.
2 Click on the **Log** tab.
3 Check or uncheck the following options: **Trace enabled**, **Database trace enabled**, and **Job execution Statistics enabled**.

 **Tip:**

- The entries for any particular job can be viewed in the Job Explorer.
- You may want to turn off **Load log on startup** option in the program **Options**. This will save some memory and slightly improve overall performance. However, you will not be able to see past log entries for the selected jobs in the Job Explorer.

**See also:**

Job Activity and System Events Logs
Log Viewer
Trace Features
Job Execution Statistics

# Changes to the System Time

The 24x7 Scheduler deals with changes to the system date/time in the following manner:

If the clock is set back (i.e. to an earlier date or time), the 24x7 Scheduler will not run any jobs until the original or rescheduled time is reached. If the 24x7 Scheduler is restarted (either manually, or because the system is restarted), it will adjust the scheduled time for the jobs that run repeatedly at the specified time interval.

If the clock is set forward (i.e. to a later date or time), the 24x7 Scheduler will run any jobs that were missed because of the time change and have the **Skip** option set to No. It will also run any jobs that have the **Skip** option set to Yes and the actual delay falls in the allowed interval. If the 24x7 Scheduler is restarted (either manually, or because the system is restarted), it will adjust the scheduled time for the jobs that run repeatedly at the specified time interval.

# Working with Job Database

The 24x7 Scheduler stores information on scheduled jobs in the **Job Database**. The **Job Database** consists of a single file with the default name SCHEDULE.DAT. By default, this file is located in the 24x7 Scheduler installation

directory. You should not attempt to edit job database files directly. If you have installed the 24x7 Scheduler in more than one directory or on two or more computers and you wish to copy information on scheduled programs between them, you can copy that file to replicate the job database. The 24x7 Scheduler also stores some configuration information in the System Registry under the registry key *HKEY_LOCAL_MACHINE\SOFTWARE\SoftTree Technologies, Inc.\24x7*. This information includes all 24x7 Scheduler configuration options available through the Tools/Options menu, definitions of database profiles, and definitions of remote agent profiles. In case of system recovery you can try modifying this information directly, but it is highly recommended to use 24x7 Scheduler GUI.

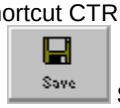
Starting with the version 1.6 24x7, Scheduler supports multiple job database files. You can create, modify and save job databases similar to the way in which you create, modify and save spreadsheet files in your favorite spreadsheet program. However, the last opened job database is always the active one. So, if you have a Windows shortcut pointing to a particular 24x7 Scheduler job, always remember to update the shortcut or reopen the correct job database. This should help avoid jobs failing.

For your convenience, 24x7 includes a **Database Manager** tool. You can use this tool to copy jobs between job databases.

On startup, the 24x7 Scheduler loads the Job Database into computer memory. This significantly improves overall job processing performance when compared with slow disk access operations required for processing jobs stored on disk. For your convenience, the loaded version of the job database is broken into two parts:

1 A binary image of the database in the disk file.
2 An active job pool that consists of enabled jobs only.

When you add a new, delete, disable, enable, or simply modify an existing job, you make changes to the binary image of the database. This is similar to when you open and edit text files in Windows Notepad. The 24x7 Scheduler does not submit your changes to the active job pool until you complete all the necessary changes. You then save those changes by doing one of the following:

- Click **File** menu, then click **Save**.
- Press shortcut CTRL S.
- Click the 

**Save** button on the toolbar.

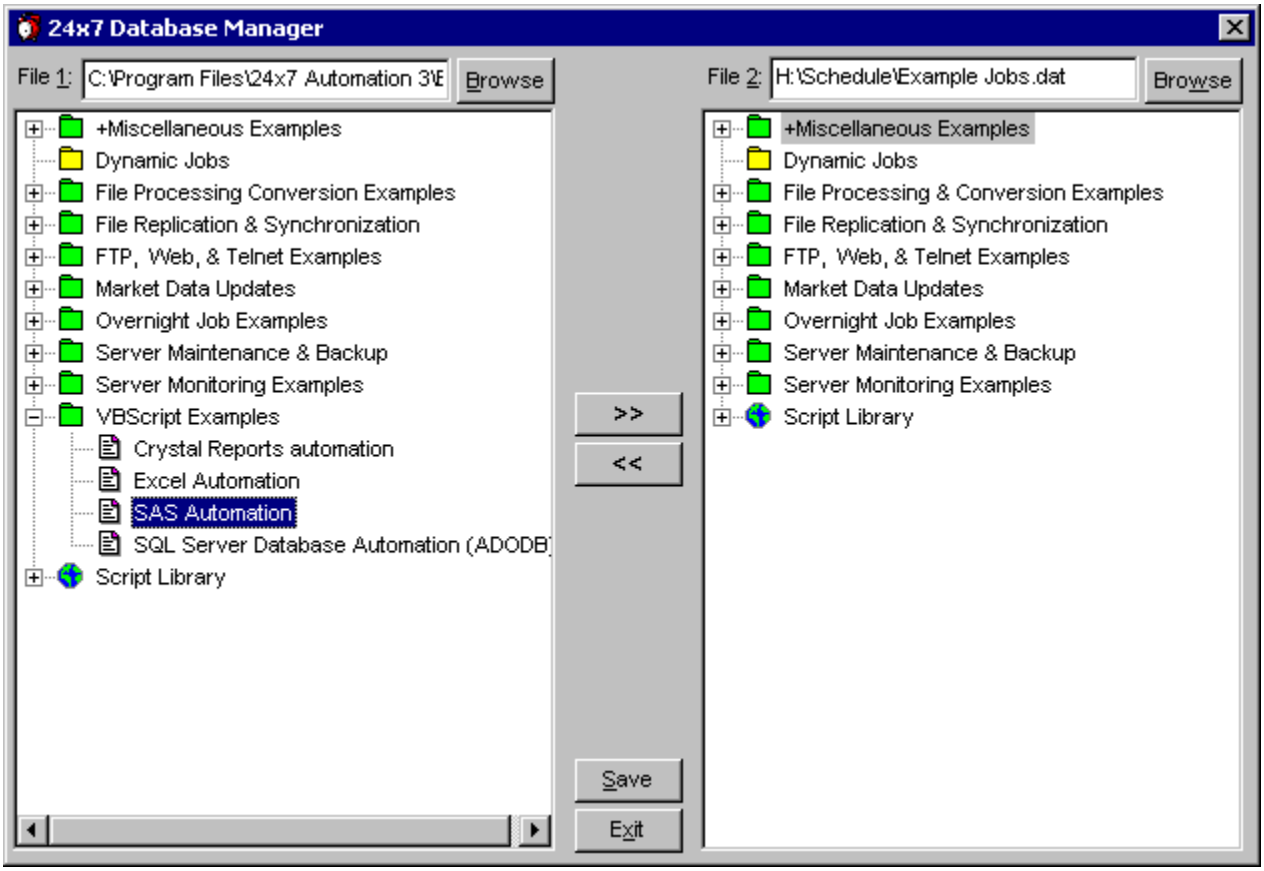
On the **Save** command, the 24x7 Scheduler saves changes in the disk file then commits updated job definitions in the active job pool.

**Tip:**

- Because there are two steps involved in the updating job definitions, you have an option of rejecting changes by restarting the 24x7 Scheduler without saving your changes first.

# Job Database Manager

You can use the Job database Manager to copy jobs and job folders between different 24x7 job database files. To start Job Database Manager click **Tools/Job Database Manager** menu.




**To copy job from/to another job database**

1 Use the left Browse button to find and open the source or target job database file.
2 Use the right Browse button to find and open another job database file.
3 In the source job database find and select the job to be copied.
4 In the target job database find and select the target folder.
5 Click the **>>** button to copy the selected job from the database displayed on the left side to the right side database or click the **<<** button to copy the selected job from the right side database to the left side database. If necessary repeat steps 3 to 5 to copy additional jobs.
6 Click the Save button to save changes.

**Important notes:**
The copied job is always added to the target job database. If the target folder already contains a job with the same name you get two jobs with the same name but different job IDs. The ID of the added is changed to a unique number in the target job database.
To delete duplicate jobs close the Job Database Manager then open the job database using File/Open menu and delete these jobs.

**Tip:**
Job Database Manager supports drag-and-drop interface that you can use to quickly copy individual jobs and entire job folders.

**To copy job folder with contained jobs from/to another job database**

1 Use the left Browse button to find and open the source or target job database file.
2 Use the right Browse button to find and open another job database file.
3 In the source job database find and select the folder to be copied.
4 In the target job database find and select the target folder.

5 Click the **>>** button to copy selected folder jobs from the database displayed on the left side to the right side database or click the **<<** button to copy selected folder jobs from the right side database to the left side database. If necessary repeat steps 3 to 5 to copy additional folders.
6 Click the Save button to save changes.

**Important notes:**
The copied jobs are always added to the target job database. The ID of each added job is changed to a unique number in the target job database.

**To copy individual Script Library statements from/to another job database**

1 Use the left Browse button to find and open the source or target job database file.
2 Use the right Browse button to find and open another job database file.
3 In the source job database find and select the statement to be copied.
4 Click the **>>** button to copy the selected statement from the database displayed on the left side to the right side database or click the **<<** button to copy the selected statement from the right side database to the left side database. If necessary repeat steps 3 to 4 to copy additional statements.
5 Click the Save button to save changes.

**Important notes:**
The copied statement is added to the target job database if there is no matching statement with the same name, otherwise it replaces the name matching statement in the target database.

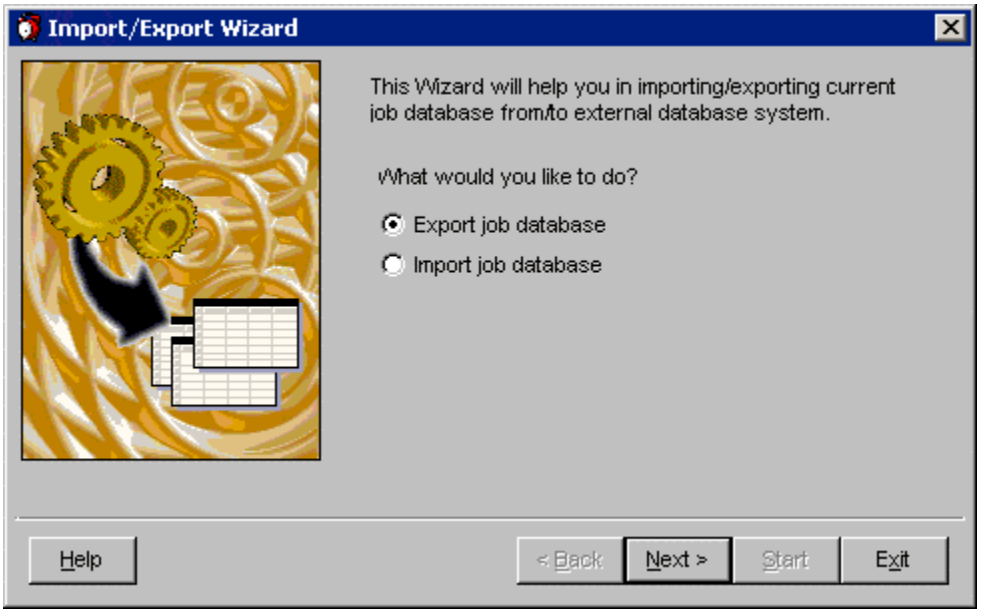
**To copy entire Script Library from/to another job database**

1 Use the left Browse button to find and open the source or target job database file.
2 Use the right Browse button to find and open another job database file.
3 In the source job database select the Script Library icon.
4 Click the **>>** button to copy the selected Script Library from the database displayed on the left side to the right side database or click the **<<** button to copy the selected Script Library from the right side database to the left side database.
5 Click the Save button to save changes.

**Important notes:**
The copied Script Library completely replaces the Script Library in the target job database.

# Job Database Import/Export Wizard

You can use the Import/Export Wizard to import and export job definitions, holiday definitions, Script Library statements and job event log to and from various relational database systems. To start Job Database Import/Export Wizard click **Tools/Import Export Job Database** menu.




The Import/Export Wizard uses an ODBC connection and supports import and export to the following database types:

- Oracle 7 and later
- Microsoft SQL Server 6 and later
- Sybase SQL Server 10 and later
- Sybase ASE
- Sybase SQL Anywhere
- Watcom SQL 5 and later
- Microsoft Access 2 and later
- Any other database system compatible on a data-type level with any database system listed above

**Note:**
You must have an appropriate ODBC driver and database profile installed and configured on your system in order to be able to import/export data using the Import/Export Wizard.

**Tip:**
Use the following guidelines when selecting database type in DBMS field:

- For Oracle and compatible database systems select ODBC**: Oracle** item
- For MS SQL Server, Sybase SQL Server, Sybase ASE and compatible systems select ODBC**: SQL Server**
- For Sybase SQL Anywhere, Watcom SQL and compatible systems select ODBC**: SQL Anywhere**
- For MS Access and compatible systems select ODBC**: MS Access**

The Import/Export Wizard uses the selection to determine data-types supported by the target database so it can properly generate table syntax when exporting and importing data.

**To export data to your database**

1 Select the Export option.
Click the Next button.
2 Check data that you want to export.
Click the Next button.
3 This step is optional and it is created only if the export jobs option is selected and job database contains password-protected jobs.
Enter passwords for password-protected jobs. Jobs left without a password entered will be ignored and not exported to the target database.
Click the Next button.
4 Select the appropriate database type in the DBMS drop-down field.
Select the appropriate ODBC profile name the Data Source drop-down field.
If required enter your database use name/ID into Login ID field.
If required enter your database password into the Password filed.

If connecting to SQL Server or ASE database enter the target database name into the Database field.
Click the Next button.

5 Click the Start button and watch the status box for progress-of-work messages and database errors if any occurs.

**To import data from your database**

1 Select the Import option.
Click the Next button.

2 Check data that you want to import.
Click the Next button.

3 Select the appropriate database type in the DBMS drop-down field.
Select the appropriate ODBC profile name the Data Source drop-down field.
If required enter your database use name/ID into Login ID field.
If required enter your database password into the Password filed.
If connecting to SQL Server or ASE database enter the source database name into the Database field.
Click the Next button.

4 Click the Start button and watch the status box for progress-of-work messages and database errors if any occurs.

# Security

For your convenience 24x7 Scheduler supports several security features that you may use to secure your jobs and restrict access to **24x7 Master Schedulers** and **24x7 Remote Agents**. These features are described in details below.

# Job Protection and Job Password

Because the data and settings you have in your jobs may be confidential, 24x7 Scheduler includes flexible security features to ensure that unauthorized people cannot access your data. **Job Protection** allows you to setup a password and access level to individual jobs.

**To protect a job:**

1 Select the desired job

2 Click File/Protected menu to setup new protection, remove existing protection or change protection level for the job. You can also right-click on the job icon and then select **Protect** item from the popup menu. The job protection dialog appears.

3 Type your password, select desired protection level, then click the OK button.

For more information about job protections see **Protecting/Unprotecting Jobs** topic.

# Job Database Security

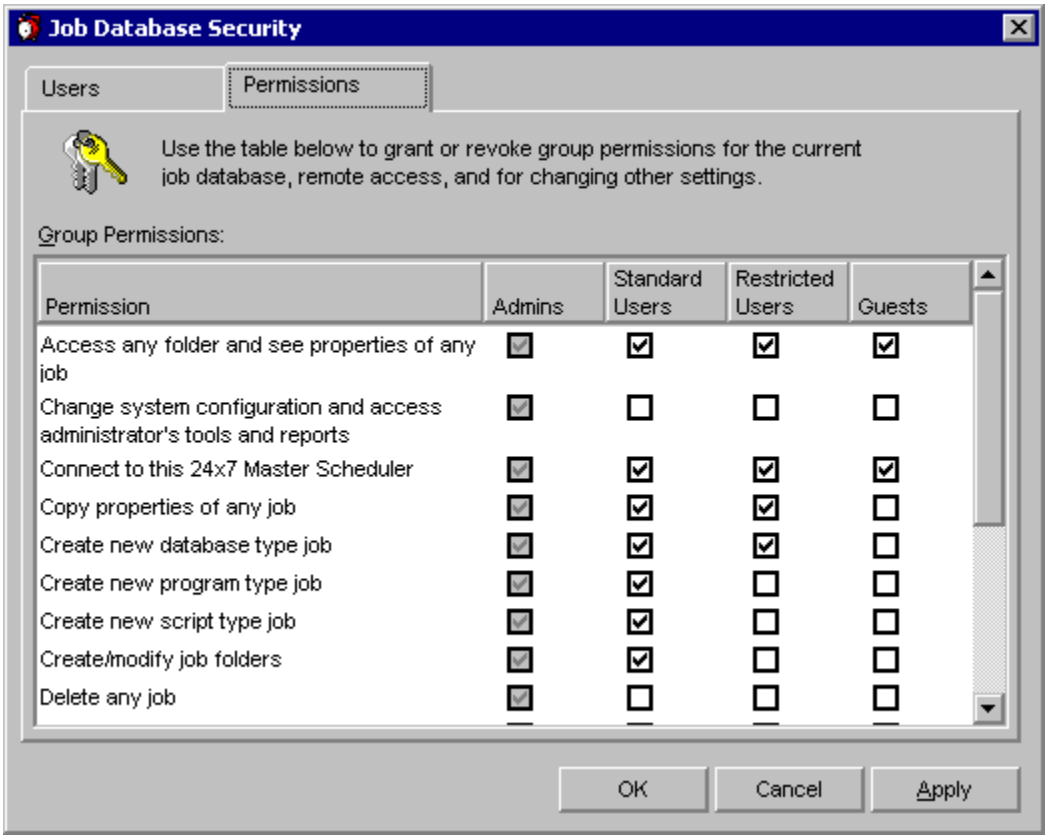
When you first install 24x7 Scheduler, job database security system is not activated by default. Job database security is applied only when users make change on the remote 24x7 Master Scheduler using **24x7 Remote Control** or using **COM+** and **Java 24x7 API methods**. Please note that the job database security is not applied when you make changes in the local job database using 24x7 Scheduler GUI. It is assumed that only system administrators have direct access to the server computer running 24x7 Scheduler. You can manage local access security by all available means of the host operation system.

Note that the job database security includes numerous settings called privileges that can be used to protect both jobs and the system configuration from unauthorized people.

There are 4 predefined user groups exist in the 24x7 Scheduler:

- Administrators
- Standard users
- Restricted users
- Guests

You can add new users to these groups and change group security settings using **Tools/Security/Job Database Users and Permissions** menu. Note that group privileges are applied to the group as a whole. You can customize group privileges as desired. However the Administrators group privileges cannot be changed and always allow full unrestricted access to all 24x7 Scheduler features.



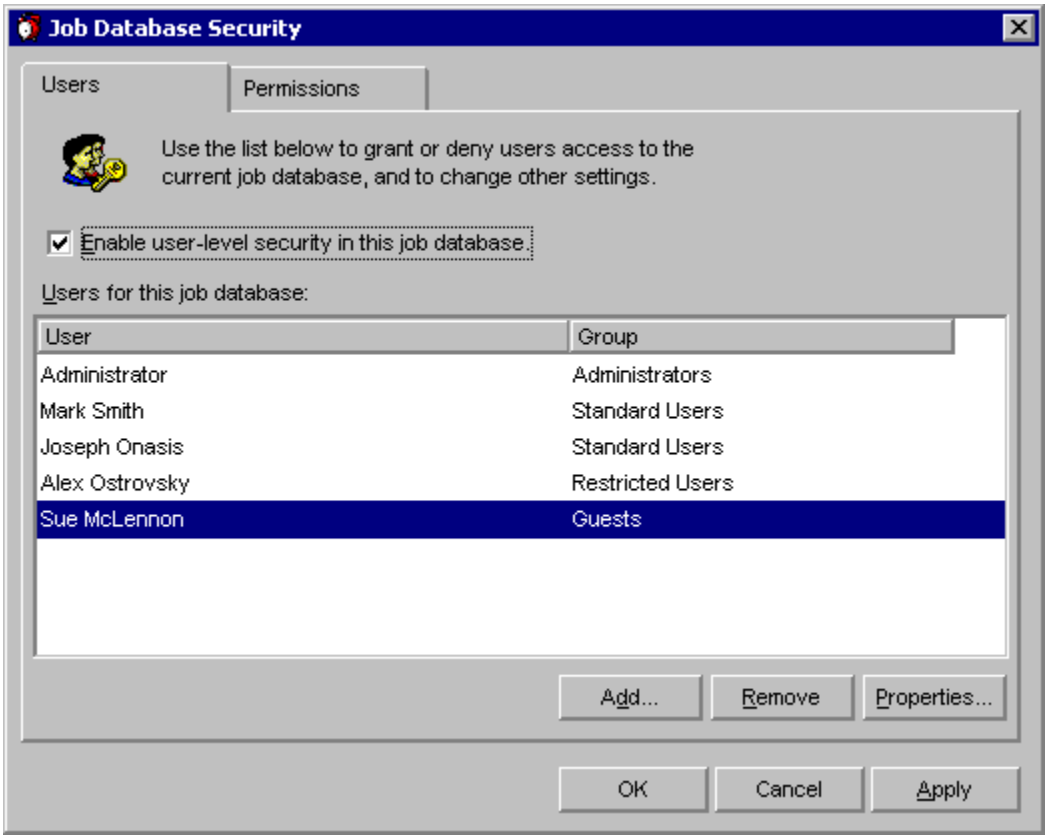

By default, members of the **Administrators** group have full access to all features; members of the **Standard Users** group have full access to all jobs and tools, but cannot change system settings; members of the **Restricted Users** group can create new **database type** jobs only and run their own jobs, as well as have "read-only" access to other jobs; members of the **Guests** group have "read-only" access to jobs and cannot create or change jobs and have no access to the system configuration settings.

**Important note:**
The job database security settings are saved within the job database file and so they remain active even after the file is copied to another computer or renamed or moved to another directory.

**To activate job database security:**

1. Select **Tools** menu, then click **Security** submenu, and then click **Job Database Users and Permissions** item. The Job Database Security dialog will appear.
2. Check **Enable user level security in this job database** box

3 Create new users using **Add** button. Follow instructions provided on the **Add/Modify User** dialog.




4 Switch to the Permissions tab page and set group permissions as required.
5 Click OK button and then click the Save button on the Job Explorer toolbar to save changes in the job database.

For complete list of supported privileges and their descriptions see **Job Database Security Privileges** help topic.

**Important Notes:**

- **If the security system is activated, users that do not exist in the job database will be denied remote access to the 24x7 Master Scheduler.**
- **You must restart 24x7 Scheduler before your new changes will take full effect!**
- For compatibility with previous versions, 24x7 Scheduler still supports **restricted access by serial number**. This can be used in addition to the methods described above.

# Remote Agent Security

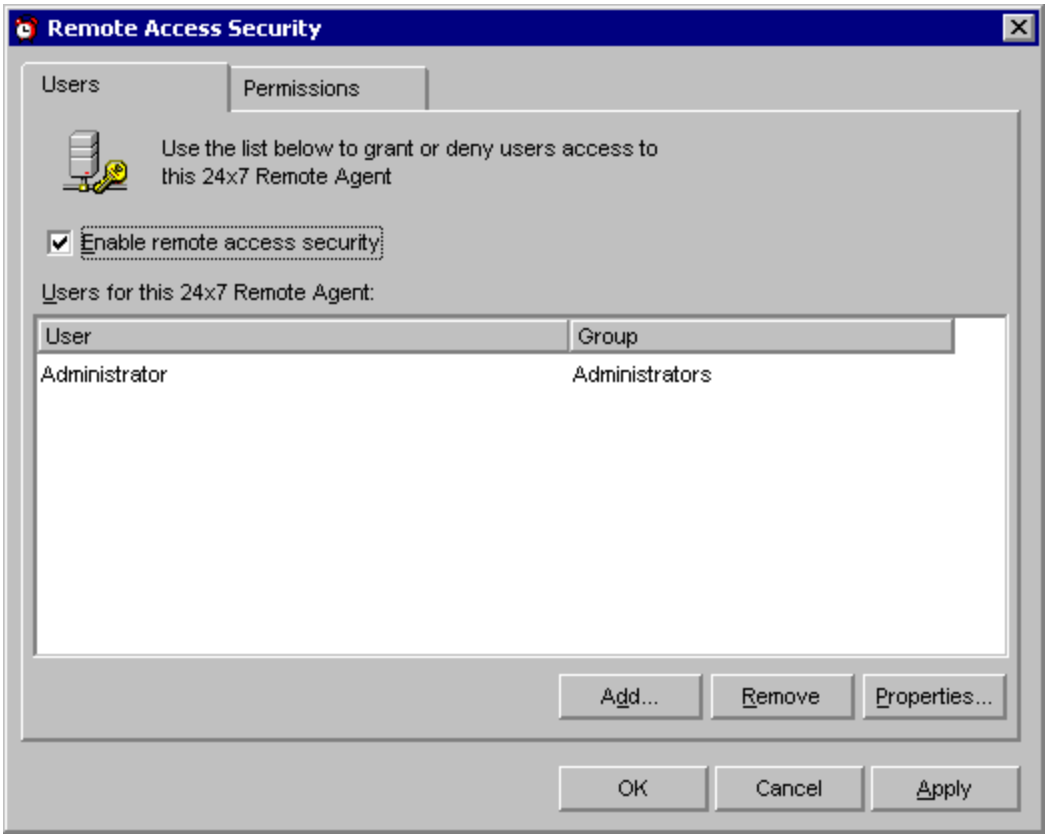
When you first install 24x7 Scheduler, Remote Agent security system is not activated by default.
Remote Agent security can be used to restrict access to the **24x7 Remote Agents**.

Agent's security is applied when users connect to the agent, run jobs or make other changes on the Remote Agent using **24x7 Remote Control** or using **COM+** and **Java 24x7 API methods**. Please note that the security is not applied when you make configuration changes locally using 24x7 Remote Agent GUI. It is assumed that only system administrators have direct access to the server computer running 24x7 Remote Agent. You can manage local access security by all available means of the host operation system.

Note that the agent's security includes numerous settings called privileges that can be used to protect the system configuration from unauthorized people and also restrict who can connect to the agent and what kind of jobs they can be submitted to the agent.

There are 4 predefined user groups exist in the 24x7 Remote Agent:
- Administrators
- Standard users
- Restricted users
- Guests

You can add new users to these groups and change group security settings using **Tools/Security/Job Database Users and Permissions** menu. Note that group privileges are applied to the group as a whole. You can customize group privileges as desired. However the Administrators group privileges cannot be changed and always allow full unrestricted access to any 24x7 Scheduler's feature.



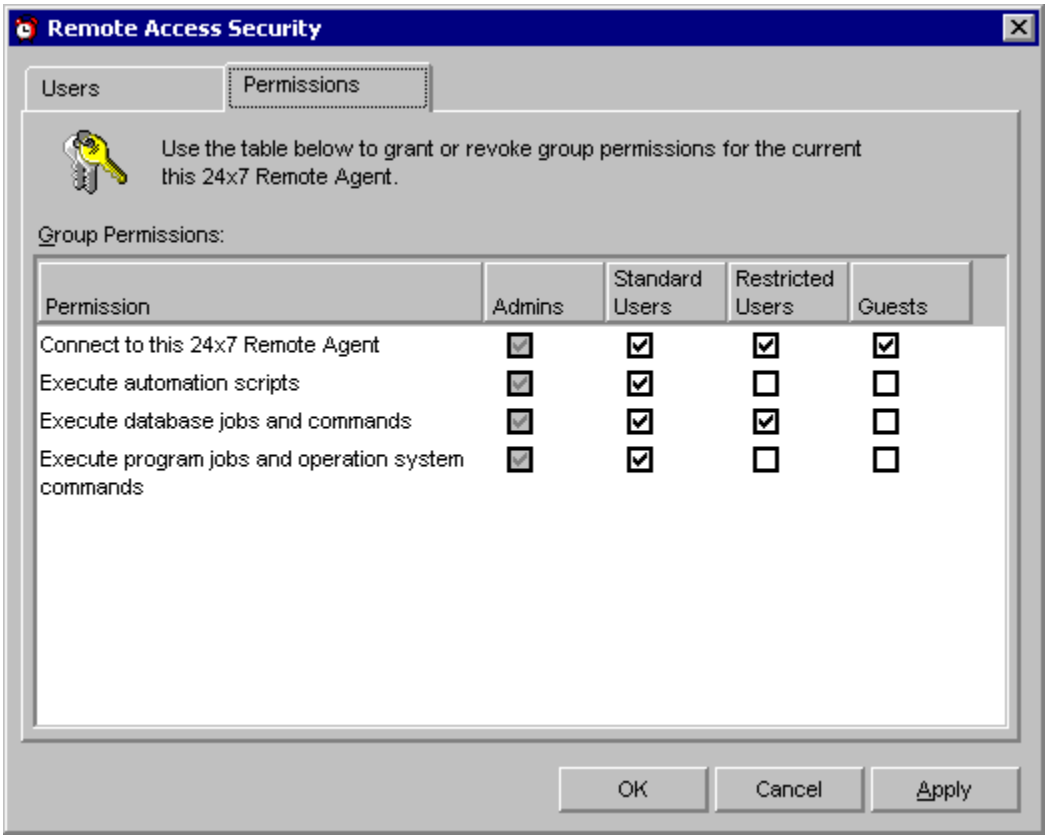

By default, members of the **Administrators** group have full access to all features; members of the **Standard Users** group can connect and execute any job, but cannot change system settings; members of the **Restricted Users** group can connect and execute **database type** jobs only, they also cannot change system settings; members of the **Guests** group cannot execute any job and have no access to the system configuration settings, they can only test connection to the agent.

**Important note:**
The security settings are saved in the RACCESS file that you should protect from accidental or unauthorized deletion using security features of the host operation system. Grant modify and delete privileges for this file only to the members of the **Administrators** group.

**To activate agent security:**

1. Select **Tools** menu, then click **Security** submenu, and then click **Remote Agent Access and Permissions** item. The Remote Access Security dialog will appear.
2. Check **Enable remote access security** box

3 Create new users using **Add** button. Follow instructions provided on the **Add/Modify User** dialog.




4 Switch to the Permissions tab page and set group permissions as required.
5 Click OK button.

For complete list of supported privileges and their descriptions see **Remote Agent Security Privileges** help topic.

**Important Notes:**

- **If the security system is activated, users that do not exist in the agent's security file will be denied remote access to the 24x7 Remote Agent.**
- Changes take effect immediately after you click the OK or Apply buttons on the Remote Access Security dialog
- For compatibility with previous versions, 24x7 Scheduler still supports **restricted access by serial number**. This can be used in addition to the methods described above.

**See also:**

Job Database Security Privileges
Remote Agent Security Privileges
Changing Password

# Job Database Privileges

A job database privilege is the right to execute a particular operation or it is the right to access another user's job. You can grant these privileges to individual user groups.

There are two types of privileges: system privileges and job privileges.

System privilege is the right to perform a particular action, or change system configuration. An example of a system privilege is the right to use Job Database Manager tool delete the rows of any table in a database.

Job privilege is the right to perform a particular action on a specific job. An example of a job privilege is the right to execute a job created by another user.

| Privilege | Description |
|---|---|
| Access any folder and see properties of any job | Grantee can see all job folders, all existing jobs and their properties including jobs created by other users. If this privilege is revoked, grantee can see owned jobs only. |
| Change system configuration and access administrator's tools and reports | Grantee has administrator level privileges and can change system configuration and use administrators' tools and reports. All items under Reports and Tools menus fall into administrator's tools category. |
| Connect to this 24x7 Master Scheduler | Grantee is allowed to connect to the 24x7 Master Scheduler. |
| Copy properties of any job | Grantee can create new jobs and copy properties of any job including jobs created by other users. If this privilege is revoked, grantee can still copy properties of owned jobs. |
| Create new database type job | Grantee can create new jobs of database type. If the grantee does not have this privilege but has "Create new script type jobs" privilege, he or she can still execute database commands using methods available in automation scripts. This privilege also ensures that grantee cannot change type of other existing job to the database type. |
| Create new program type job | Grantee can create new jobs of program type. If the grantee does not have this privilege but has "Create new script type jobs" privilege, he or she can still run programs using methods available in automation scripts. This privilege also ensures that grantee cannot change type of other existing job to the program type. |
| Create new script type job | Grantee can create new jobs of script type. This privilege also ensures that grantee cannot change type of other existing job to the script type. |
| Create/modify job folders | Grantee can create new and modify names and descriptions of existing job folders. |
| Delete any job | Grantee can delete any job including jobs created by other users. If this privilege is revoked, grantee can still delete owned jobs. |
| Disable/enable any job | Grantee can enable or disable any job including jobs created by other users. If this privilege is revoked, grantee can still enable or disable owned jobs. |
| Modify any job | Grantee can modify any job including jobs created by other users. If this privilege is revoked, grantee can still modify owned jobs. |
| Monitor jobs and job queues | Grantee can use Job Monitor and Queue Monitor tools. Note that in the current version remote queue monitoring is not yet supported but it may be supported in future. |
| Move any job between job folders | Grantee can move any job to a different job folder. This also includes jobs created by other users. If this |

| | |
|---|---|
| | privilege is revoked, grantee still can move owned jobs. |
| Protect/unprotect any job | Grantee can protect and unprotect any job including jobs created by other users. If this privilege is revoked, grantee still can protect and unprotect owned jobs. In order to unprotect a job, grantee must know the valid job password, without that password a job cannot be unprotected. |
| Run/debug any job | Grantee can run and debug any job including jobs created by other users. If this privilege is revoked, grantee still can run and debug owned jobs. |
| View/modify job dependencies | Grantee can access Job Dependencies Editor, which he or she can use to view and modify job dependencies including dependencies for jobs created by other users. If this privilege is revoked, grantee cannot access Job Dependencies Editor, but still can specify semaphore files in the job properties. |

**See also:**
Security
Remote Agent Security Privileges

# Remote Agent Privileges

A remote agent privilege is the right to execute a particular operation. You can grant these privileges to individual user groups.

There is only one type of privileges: system privileges.
System privilege is the right to perform a particular action, or change system configuration. An example of a system privilege is the right to use change system configuration.

| **Privilege** | **Description** |
|---|---|
| Connect to this 24x7 Remote Agent | Grantee is allowed to connect to the Remote Agent. |
| Execute automation scripts | Grantee can execute jobs having script type and also execute anonymous automation script via using 24x7 Remote Control API. |
| Execute database jobs and commands | Grantee can execute jobs having database type. If the grantee does not have this privilege but has "Execute automation scripts" privilege, he or she can still execute database commands using methods available in automation scripts. |
| Execute program jobs and operation system commands | Grantee can execute jobs having program type and also execute operation system command by using 24x7 Remote Control API. If the grantee does not have this privilege but has "Execute automation scripts" privilege, he or she can still execute database commands using methods available in automation scripts. |

**See also:**
Security
Job Database Security Privileges

# Changing Password

There exist two ways to change a password for an existing user: you can connect yourself to the remote system using **24x7 Remote Control** and use **Tools/Security/Change Password** menu to change your password or you can ask your system administrator to modify your user's settings and change the password for you. In the last case, the system administrator can use **Tools/Security/Job Database Users and Permissions** or **Tools/Security/Remote Agent Access and Permissions** menus to change your password. The menu choice depends whether the password is changed for the 24x7 Master Scheduler or 24x7 Remote Agent. In both cases the system administrator can make this change either locally on the target computer or remotely using 24x7 Remote Control.

# UNIX Remote Automation Server Security

For information on 24x7 Remote Automation Server for UNIX see **24x7 RAS for Linux and UNIX Manual**

# Sending and Receiving E-mail Messages

# Overview

24x7 Scheduler supports three e-mail interfaces: standard Windows MAPI (Messaging Application Programming Interface) interface, SMTP (Simple Mail Transfer Protocol), and Lotus Notes interface utilizing Lotus Notes API. You select the desired interface in the system options. 24x7 Scheduler uses the selected interface when sending various notification messages and executing JAL (Job Automation Language) mail statements. Both MAPI and Lotus Notes interfaces support sending and receiving simple e-mail messages as well as e-mail messages that contain one or more attachments. 24x7 Scheduler supports unlimited number of attachments in a single message. However, the maximum number of attachments can be limited by your e-mail system. Please read your e-mail system documentation on this subject. 24x7 Scheduler does not support attachments of the type "embedded OLE object". SMTP interface supports sending e-mails only. When reading incoming e-mail messages, 24x7 Scheduler, by default, saves all attached files in the temporary directory. The location of the temporary directory specified by the TEMP environment variable. Make sure you delete these files from the temporary directory when you no longer need them.

 **Important Notes:**

- MAPI enables the 24x7 Scheduler to interact with multiple messaging systems across a variety of software and hardware platforms whereas Lotus Notes interface works with the following configurations only:
  1) Lotus Notes workstation v4.5 and above running on Windows NT (NT/2000/XP/2003/VISTA/2008/7) workstation or server (Intel platforms only). There is no limitation on the Lotus Notes server version and platform.
  2) Lotus Notes server v4.5 and above running on Windows NT (NT/2000/XP/2003/VISTA/2008/7) server (Intel platform only).
- If you have installed Lotus Notes MAPI extensions, you can still use MAPI interface to send and receive e-mail via Lotus Notes. We highly recommend using MAPI interface whenever possible, because (due to nature of Lotus Notes API) we cannot guarantee that Lotus Notes interface will be compatible with the future versions of Lotus Notes. However, we will try to release updated versions of Lotus Notes interface once new versions of Lotus Notes become available.
- For SMTP interface, the 24x7 Scheduler currently supports sending e-mail with or without attachments only. It does not support receiving e-mails. If you setup an "e-mail watch" job, 24x7 Scheduler will use the MAPI interface to check for incoming e-mails.

# About Lotus Notes interface

The 24x7 Scheduler Lotus Notes interface consists of two parts:

- Notes e-mail interface library
- Notes extension manager

The 24x7 Scheduler e-mail interface library for Lotus Notes always uses default Notes mail database and mail server. Default settings are taken from Notes environment variables. When searching for incoming messages, 24x7 Scheduler always checks unread messages from the "Inbox" view.

**Warning:**
The 24x7 Scheduler installs the Notes extension manager program. This program will intercept the Notes password prompt and supplies the password that you specified for e-mails in the 24x7 Scheduler. The extension manager will allow you to send and receive e-mail messages via Notes without you having to intervene. The extension manager program is built as a set of dynamic link libraries (DLLs). These DLLs are loaded by Lotus Notes on startup, and they behave as if they are part of the Lotus Notes software. While the 24x7 Scheduler e-mail operation is in progress, your Lotus Notes is exposed to other users and programs. This is because no password is needed when interacting with Lotus Notes software. **Before selecting Lotus Notes interface, make sure you do not have another Notes extension manager already installed on your system.** To check this, make sure you do not have EXTMGR_ADDINS key in the NOTES.INI file or that key is not initialized.

# Maintenance for Holidays, Database Profiles, and Remote Agents

The 24x7 Scheduler allows you to define some information that can be shared by all scheduled jobs. This information is stored outside of the job definition database.

The following information is shared:
- Holiday list,
- Database Profile definitions,
- Remote Agent definitions.

The holiday table is stored in the HOLIDAY.TXT file. This file is loaded up by 24x7 Scheduler on startup. Definitions of database profiles and remote agents are stored in the Windows system registry. It is highly recommended that you do not modify the registry directly, but instead maintain this information using 24x7 Scheduler utilities that can be found under **Tools** menu. You should update this information on an as-needed basis. For example, when the database password expires, you should update all database profiles using this password, otherwise all jobs utilizing outdated information will fail.

**See also:**

Holiday List

Database Profiles

Remote Agent Profiles

# Chapter 2: Installation and Uninstallation

## Installation

The 24x7 Scheduler's Setup program provides three installation modes:

- Typical,
- Compact,
- Custom.

In the "Typical" mode, Setup installs all available components by default. As a result you will not need to answer too many questions.

In the "Compact" mode, Setup installs 24x7 Scheduler only and does not install database drivers, example jobs and other auxiliary files.

In the "Custom" mode, Setups offers a choice of components to be installed. You will need to check/uncheck components to be installed.

If you are not going to use the database features incorporated in 24x7 Scheduler, you will not need to install all of the database support. You will be able to go ahead with the "Compact" installation.

If you will be using ODBC interface, configure the data sources you will need using the ODBC Administrator.

## VBScript Debugger Dependencies

**24x7 Debugger depends on certain files that are installed with Microsoft Internet Explorer 4.0 or better. Make sure you have Internet Explorer installed before you attempt to debug a VBScript job.**

## Windows NT (NT/2000/XP/2003/VISTA/2008/7) Specifics

Before installing the program, make sure that you are logged-on using an account that is a member of the local Administrators group.

Optionally, the 24x7 Scheduler can be setup to run as a Windows NT (NT/2000/XP/2003/VISTA/2008/7) system service.

As always, there are both advantages and disadvantages associated with running the 24x7 Scheduler as a service rather than as a standalone application. The advantages include the following:

- The 24x7 schedule service can be configured to start automatically whenever the computer is started and it can run continuously in the background, regardless of whether a user is logged on or not.
- Programs run by the 24x7 service inherit the security attributes of the 24x7 service. This gives them the same security permissions as the operating system.
- Due to Windows NT (NT/2000/XP/2003/VISTA/2008/7) security restrictions, the 24x7 Scheduler, while running as a service, is not able to interact with the Desktop. So you can't do much with it other than stop and start the service. Since 24x7 Scheduler is normally setup to start automatically whenever the computer is restarted, you will not normally have any dealings with it. Instead, you should use the standalone version of the 24x7 Scheduler to manage scheduled jobs.
- You will not be able to see any error or warning messages on the screen. However, if you enable the "Logging to the system event log" option, you will be able to use the Windows NT Event Viewer to see these messages in the Windows NT system event log.

You can reconfigure the 24x7 schedule service to a certain extent using the Control Panel services applet. Before you make any changes, make sure you understand the Windows NT service concepts. You should refer to the following Microsoft knowledge base articles for more information:

Q124184 - Service Running as System Account Fails Accessing Network.
Q132679 - Local System Account and Null Sessions in Windows NT.
Q152451 - Applications Run from the Schedule Service Fail to Print.

To enable or disable the Windows NT (NT/2000/XP/2003/VISTA/2008/7) service, carry out the following:
1 Select the **Tools/Options** menu. The **Options** dialog will appear.
2 Select **Service** tab page.
3 Check "Run as a Windows NT service" .
4 Specify the desired settings for the 24x7 Scheduler Windows NT service.
5 Click the **OK** button.
6 Restart your computer.


**You must either restart your computer or manually restart the 24x7 service before new service related settings would take effect.**


**To install 24x7 Automation Suite on a Windows Server through Terminal Services:**
1 Set up a Windows 2000/XP/2003 Server computer as a Terminal Services Host and configure it to allow both administrator and non-administrator access.
2 Run the installation program on this machine logged on as an Administrator. The 24x7 Automation Suite should be installed in Terminal Services' Install mode. Use the command line "change user /install" or use Add/Remove Programs in the Control Panel. This is required so that Terminal Services can appropriately manage the registry for each user.


In order to make behavior of 24x7 Scheduler and accompanying products installed on Windows Terminal Server consistent with other installations, the 24x7 Automation Suite creates HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Terminal Server\Compatibility\Applications\24X7 value, which consists of the following flags:
- Windows 32-bit application: 0x00000008
- Disable registry mapping for application: 0x00000100
- Do not substitute user Windows directory: 0x00000400

# Windows 9x (95/98/Me) Specifics

Optionally the 24x7 Scheduler can be setup to run as a Windows 95/98/Me service application. This option allows 24x7 Scheduler to start automatically whenever the computer is started and it can run continuously in the background, regardless of whether a user is logged-on or not.

To enable or disable the Windows 95/98/Me service application:
1 Select the **Tools/Options** menu. The **Options** dialog window will appear.
2 Select **Service** tab page.
3 Check "Run as a Windows 95/98/Me service" .
4 Specify the desired settings for the 24x7 **schedule service**.
5 Click the **OK** button.
6 Restart your computer.

# Installing Remote Agents

24x7 Remote Agents for Windows are installed as an integral part of 24x7 Automation Suite and cannot be installed separately. See Remote Agents topic for more information on configuring Remote Agents.

24x7 Remote Automation Server for UNIX and Linux are installed using special installation scripts that are provided only to registered users.

# Automating the Installation Process; Silent Setup

If you plan to install the 24x7 Scheduler using the same installation settings on a large number of computers, you can record the installation process on one computer and then play it back on the others. This allows you to run the Setup program on the other computers automatically, without user input ("silent" installation).
A silent installation does not prompt the user for input.

The Setup program accepts optional command line parameters. These can be useful to system administrators, and to other programs calling the Setup program.

**/SP-**
Disables the This will install... Do you wish to continue? prompt at the beginning of Setup.

**/SILENT, /VERYSILENT**
Instructs Setup to be silent or very silent. When Setup is silent the wizard and the background window are not displayed but the installation progress window is. When a setup is very silent this installation progress window is not displayed. Everything else is normal so for example error messages during installation are displayed and the startup prompt is

If a restart is necessary and the '/NORESTART' command isn't used (see below) and Setup is silent, it will display a Reboot now? message box. If it's very silent it will reboot without asking.

**/NORESTART**
Instructs Setup not to reboot even if it's necessary.

**/LOADINF="filename"**
Instructs Setup to load the settings from the specified file after having checked the command line. This file can be prepared using the '/SAVEINF=' command as explained below.

Don't forget to use quotes if the filename contains spaces.

**/SAVEINF="filename"**
Instructs Setup to save installation settings to the specified file. Don't forget to use quotes if the filename contains spaces.

**/DIR="x:\dirname"**
Overrides the default directory name displayed on the Select Destination Directory wizard page. A fully qualified pathname must be specified.

**/GROUP="folder name"**
Overrides the default folder name displayed on the Select Start Menu Folder wizard page.

**/NOICONS**
Instructs Setup not to create any icons.

**/COMPONENTS="comma separated list of component names"**
Overrides the default components settings. Using this command line parameter causes Setup to automatically select a custom type.

**/NOWSH**
Overrides the default setting to install or update Windows Scripting Host

**/SERVICE <account> <password> [agent]**
Instructs Setup to automatically install 24x7 Scheduler service. The service is not installed by default. To install the service specify the full service account name including domain for user authentication and password. To authenticate user on the local computer specify dot. For example: 247setup.exe /service .\Administrator adminpass.
If account name or password includes spaces enclose it in double quotes. The **/service** option can be used along with other setup options, however it must be specified as the last option following by the account and password and

optional "agent" word. If the "agent" is specified the service is installed and configured to run in the 24x7 Remote Agent mode.

**Tip:**

- If you are installing 24x7 Scheduler on several servers, you may want to copy the same configuration information to all of them.

  **To copy configuration information:**

  1 Install the 24x7 Scheduler on the first computer, recording the installation as described above.
  2 Configure the 24x7 Scheduler using the **Tools/Options** menu.
  3 Using the Registry Editor (REGEDIT.EXE), export the *HKEY_LOCAL_MACHINE\SOFTWARE\SoftTree Technologies, Inc.\24x7Scheduler* registry key to a file with the extension .REG. To create an export file locate the key in the Registry Editor then click **Registry/Export Registry File** menu.
  4 Copy this .REG file to the directory that contains the 24x7 Scheduler setup files.
  5 Use the Registry Editor on the destination computer to import registry keys from the .REG file. To import the file interactively use **Registry/Import Registry File** menu. To import the file silently run *REGEDIT <reg file here>* command from the DOS command prompt. To suppress "Import…" message box use the /s command line switch. Example: *REGEDIT.EXE /s 24x7.reg*
  6 Restart the 24x7 Scheduler on the destination computer.

# Uninstallation

The 24x7 Scheduler supports a standard uninstallation mechanism for removing program files from the computer.

To uninstall the 24x7 Scheduler:

1 Click Windows **Start** button, from the Start Menu select **Settings**, then **Control Panel**.
2 Double-click **Add/Remove Programs**. Click the button to launch this function.
3 Select the **24x7 Scheduler** item in the programs list, click the **Add/Remove** button
4 Delete all left files from the 24x7 Scheduler home directory. Delete the home directory.

**See also:**

System Requirements

# System Requirements

**Minimum Hardware Requirements**

1 Intel-based machine running one of the following operating system:
- Windows 95 with IE 4.0 or better
- Windows 98
- Windows Me
- Windows NT 4.0 (server or workstation) with IE 4.0 or better
- Windows 2000 (server or workstation)
- Windows XP (server or workstation)
- Windows 2003 server
- Windows Vista
-

1 16 MB disk space
2 VGA or better monitor

**Recommended Configuration**

Pentium class CPU 500 MHz or better
1 512 MB RAM or better

2 SVGA 256-color or better monitor

**Server Mode**

1 Pentium class CPU 150 MHz or better
2 At least 128 MB RAM. For each Standby Scheduler add 4MB on the server machine
3 VGA monitor

**Software Requirements**

Due to the communication mechanisms between Master Schedulers and remote Standby Schedulers and between Remote Agents, several standard Windows operating system services/protocols must be installed and running:

1 Remote Procedure Call (RPC) Service
2 Windows NT (NT/200/XP/2003/Vista/2008) only: Event Logging Service
3 MAPI interface (required to support the 24x7 Scheduler ability to process "e-mail watch" jobs and send e-mail notifications using MAPI email interface)
4 Depending on the selected communication protocol for Fail-over Mode and Remote Agents, one of the following will also be needed:
   - Winsock
   - TCP/IP
   - Named Pipes
5 Required database client software (if needed for scheduled jobs)
6 ODBC (if needed for scheduled jobs)
7 FTP interface (if needed for scheduled jobs). The following Microsoft DLLs must be installed in the Windows system directory: WININET.DLL, SHLWAPI.DLL

**See also:**

Installation and Uninstallation

# Chapter 3: Database Interfaces

The 24x7 Scheduler uses database profiles when connecting to various databases. You must define at least one database profile for each database you want the 24x7 Scheduler to connect to. A profile is set of database connection parameters that the 24x7 Scheduler uses when connecting to your database.

**Database Profiles Dialog Box**

### Description

The database profiles dialog box lists the installed database interfaces and configured database profiles defined for each interface. It will enable you to create, edit, test, and delete database profiles. In addition, you will be able to access the Configure ODBC dialog box and create, edit, or delete an ODBC data source definition.

You will find it easy to display or hide the list of configured profiles for a particular interface.

- To display or hide the list of installed database interfaces, double-click **Database Interfaces**.
- To display the list of database profiles defined for a particular interface, click the plus sign (+) preceding the interface name or double-click on the interface name itself.
- To hide the list of database profiles for a particular interface, click the minus sign (-) preceding the interface name or double-click on the interface name itself.

### New

This will display the Database Profile Setup dialog box for the selected interface and create a new database profile. Select an interface name and click **New** to display the Database Profile Setup dialog box for that interface.

### Edit

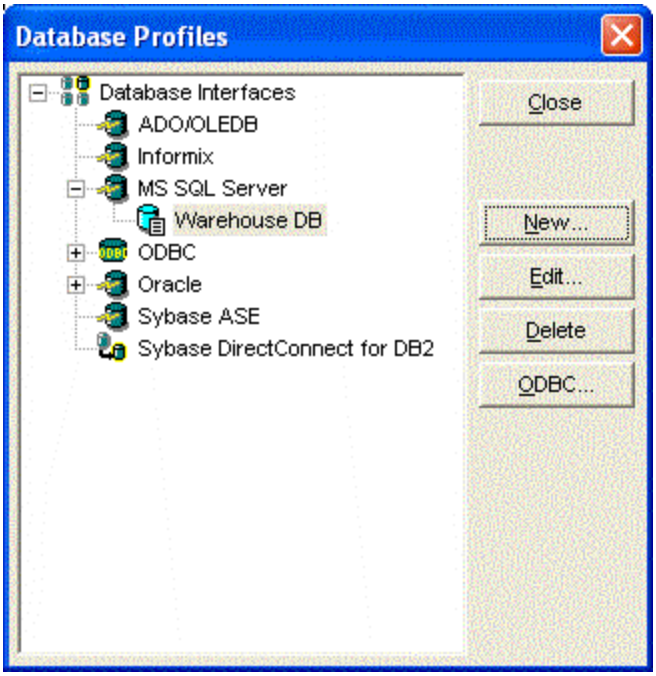
This will display the Database Profile Setup dialog box for the selected profile. You will be able to use this dialog to edit the profile. Select a database profile name and click **Edit**. This will display the Database Profile Setup dialog box for that profile.

### Delete

This will delete the selected database profile from the Database Interfaces list. Select a database profile name and click **Delete.** This will remove it from the interface list.

### ODBC

This function launches the ODBC Driver Manager and displays the Configure ODBC dialog box, which will enable you to create, edit, or delete an ODBC data source definition. For details see ODBC Driver Manager help topics.



**See also:**
Database Jobs
ODBC Interface
Database Profiles

# Database Profiles

**Database Profile Setup**
In the Database Profile Setup dialog you can edit database connection options for the chosen database profile.

The database profile setup includes the following connection options. You must supply these in order to access your database:

- **Profile Name** - The name of your database profile.
- **Provider** – This parameter is used with ADO /OLEDB database interface only. Specify name of OLE DB provider that can be used to connect to the database server. For example, for Microsoft SQL Server, the provider name is SQLOLEDB. Consult your database server documentation for details on provider name.
- **Database** - The database name. You can leave this property blank if your database does not have a name.
- **Server** - The server name or server connect descriptor specifying parameters that your database driver uses to connect to the database server. For ODBC data sources, specify the data source name.
- **Login ID** - A valid login ID for your database server.
- **Password** - The login password of your database server. The password you type in will be displayed as asterisks (*).
- **Isolation Level** - For those DBMS and database interfaces that support the use of lock values and isolation levels. The Lock preference sets and activates the isolation level when connecting to the database. In multi-user databases, transactions initiated by different users can overlap. If these transactions access common data in the database, they can overwrite each other, or collide. To prevent concurrent transactions from interfering with each other and compromising the integrity of your database, certain DBMS allow you to set the isolation level when you connect to the database. Isolation levels are defined by your DBMS, and specify the degree to which operations in one transaction are visible to operations in a concurrent transaction. Isolation levels determine the way in which your DBMS isolates or locks data from other processes while it is being accessed. You can leave this property blank if your database does not require it.
- **AutoCommit** - For those DBMS and database interfaces that support it, AutoCommit controls whether 24x7 Scheduler issues SQL statements outside or inside the scope of a transaction. When AutoCommit is unchecked, 24x7 Scheduler issues SQL statements inside the scope of a transaction. When AutoCommit is checked, 24x7 Scheduler issues SQL statements outside the scope of a transaction.
- **Asynchronous Operations** - When checked, this will allow you to perform asynchronous operations on your database. Unlike synchronous calls (which force the database client software to wait until processing is completed), asynchronous calls free the client (and 24x7 Scheduler in turn) to do other work while the server handles requests. Not all databases support asynchronous operations.
- **Number of Seconds to Wait** - When you turn on Asynchronous Operations you can specify the number of seconds you want 24x7 Scheduler to wait for a response from the DBMS. If this parameter is set to zero, the 24x7 Scheduler waits indefinitely for a DBMS response. In other words, the request never runs out of time. If the Number of Seconds to Wait value expires before the response from the DBMS, your request is automatically canceled. If your database supports asynchronous processing, you can use this parameter to prevent run-away queries.

**Testing the connection to your database**

Use the **Connect** button to test your profile settings. The Connect button connects, then immediately disconnects, the 24x7 Scheduler from the database. A connection status message is displayed after this operation.

 **Important Notes:**

- You must have database client software installed on your computer before you can establish a database connection.

- In some databases, such as SQL Server, you must turn on the AutoCommit in order to run stored procedures executing SQL DDL statements. DDL statements include: CREATE INDEX, SELECT INTO temporary table, DROP TABLE and so on.

**See also:**

Database Jobs
Database Interfaces
ODBC Interface
Testing Connection to Your Database

# ODBC Interface

The 24x7 Scheduler Open Database Connectivity (ODBC) interface accesses any ODBC data source for which you have installed an ODBC-compliant driver. This interface communicates with your driver through the ODBC Driver Manager to access the data you need.

The 24x7 Scheduler supports access to a data source through any Level 1 or higher 32-bit ODBC 1.x, 2.x or 3.0 driver obtained from SoftTree Technologies or from another vendor. In most cases, other drivers will work with the 24x7 Scheduler. However, SoftTree Technologies has not tested all drivers on the market and is unable to verify whether they will work with the 24x7 Scheduler.

You can start the ODBC Driver Manager from the Control Panel or, alternatively, you can click the **ODBC** button on the **Database Interfaces** dialog box

**See also:**

Database Jobs
Database Interfaces
Database Profiles

# Chapter 4: Scheduling Jobs

## Job Wizard

The Job Wizard is the tool you use to simplify the process of scheduling a new job or updating properties of an existing job. The Job Wizard consists of a series of dialog windows. The Job Wizard asks you questions and then, using your answers, updates the job properties. The 24x7 Scheduler uses different job properties for the different job types. The Job Wizard shows only properties appropriate for the job type you have selected.

You can change job properties, including job type, at any time. Use the **Next** and **Back** buttons displayed at the bottom of the Job Wizard window to move between the property pages. You can apply changes and close the Job Wizard at any time by clicking the **Finish** button. To cancel changes, click the **Cancel** button. Alternatively, you can use the following keyboard shortcuts:

- **Next** - ALT+N or CTRL+TAB
- **Back** - ALT+B or SHIFT+CTRL+TAB
- **Finish** - ALT+F
- **Cancel** - Escape or ALT+C


**Tips:**

- You can enter and modify SQL commands and 24x7 scripts directly in the edit box of the Job Wizard window. To edit scripts using a full-featured Editor with syntax highlighting, click the **Edit** button, next to the edit box.
- To copy and paste text while in the one of editable property fields, right-click on the field, then select the desired command from the pop-up menu.

**See also:**

Using SQL, JAL and VBS Editors

# Job Processing Flow

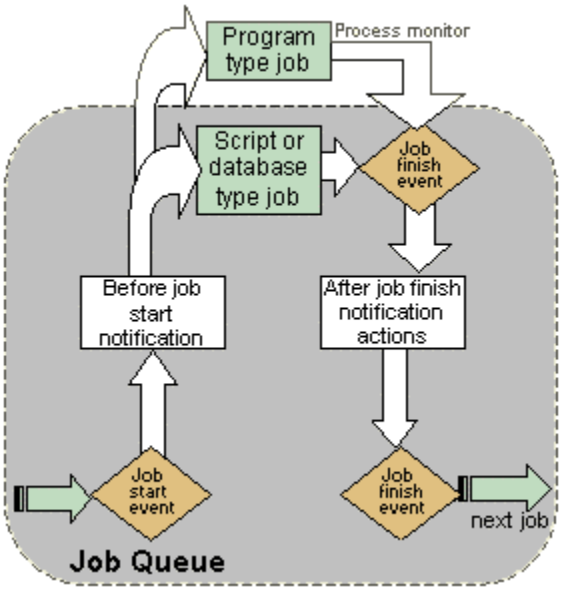
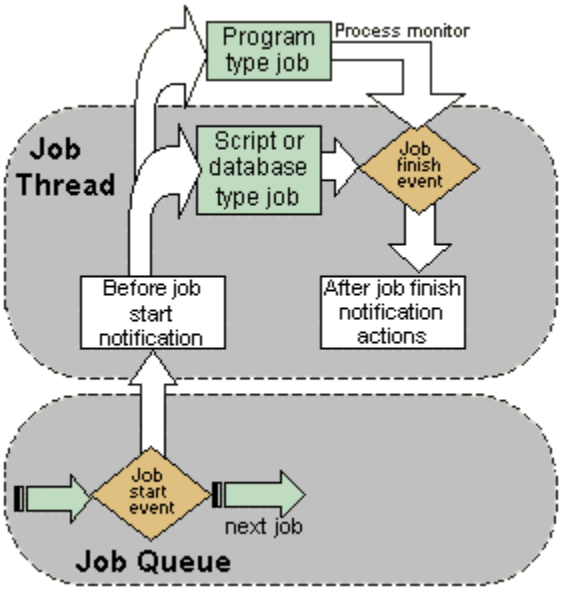
24x7 Scheduler provides maximum possible flexibility for scheduling and executing various jobs. Jobs can run in background (in other words, **asynchronous jobs**) and foreground (**synchronous jobs**); **queued** and **immediately**; normal and **detached**; **local** and **remote**. This topic outlines various options for job processing. For more information about job queues, see Job Queues topic.

Most jobs are queued before they are executed. According to the **job priority** a job can be added to the end, middle, or beginning of the assigned job queue. However, there are several exceptions to this rule. Jobs having "on startup", "on system shutdown" and "on user idle" triggers are executed **out-of-queue** because they cannot wait for the queue to become available and must be executed immediately. Also, when you use the Start Now command to test a job, that job is also executed immediately and it also runs out-of-queue. Note that these out-of-queue jobs do not interrupt or explicitly affect already running queued jobs. If there are multiple jobs that must be executed in foreground and out-of-queue, the job engine executes them sequentially one by one. The execution order for such jobs is not determined, but in most cases they are executed in their ID orders, e.g. jobs with lowest IDs are executed first. Jobs launched from other jobs are executed as usual, they are added to the queue according to their priority and the assigned queue.

 **Important Notes:**

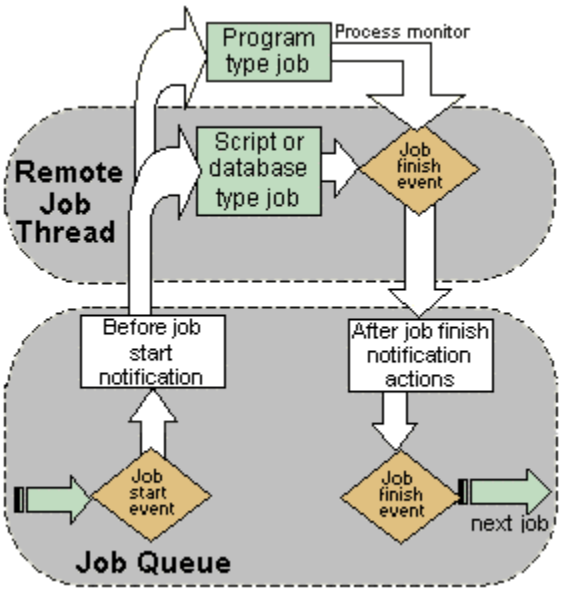
- Multiple job queues operate independently and concurrently.
- Jobs are added to their queue as soon as their start conditions are met.
- Asynchronous jobs require more system resources to run. A new job thread must be allocated for every asynchronous jobs and a separate instance of the job engine and various utilities must be loaded into computer memory for each asynchronous job run.
- It is recommended that whenever possible you should schedule synchronous jobs. Configure multiple job queues if you need to run multiple concurrent jobs.
- Jobs that tend to leak system resources or cause problems for other jobs should be setup to run **detached**. A detached job is executed in a separate system process. When that process is complete, all allocated resources are forcibly returned back to the operation system.
- You can use **semaphore files** and **global variables** to built various job dependencies and synchronize jobs running in different job queues.
- You cannot gracefully exit 24x7 Scheduler until all remote jobs are complete. The scheduler will remain in shutdown phase until all remote jobs get completed. New jobs are not started while the shutdown is in progress. Use the Windows Task Manager to forcedly terminate 24x7.exe if you must stop the processing immediately and cannot wait for remote jobs.

The following table describes processing flow within a job queue for locally executed synchronous and asynchronous jobs.

<table>
<tr><th colspan="2">Synchronous local job processing</th></tr>
<tr><td> 
</td><td>
<ul>
<li>Queued job is launched and monitored. The job queue manager waits for the job to complete before processing next queued job.</li>
<li>New jobs can be added to the queue while this job is running.</li>
<li>New jobs cannot be started in the same queue while this job is running. However, new jobs can be started in other queues while this job is running.</li>
<li>New jobs cannot be also started while the notification actions (if any) are being executed.</li>
<li>Already running jobs keep running uninterrupted.</li>
</ul>
</td></tr>
<tr><th colspan="2">Asynchronous local job processing</th></tr>
<tr><td> 
</td><td>
<ul>
<li>Queued job is launched and monitored.</li>
<li>New jobs can be added to the queue while this job is running.</li>
<li>New jobs can be started in the same job queue while this job is running.</li>
<li>New jobs can be also started while the notification actions (if any) are being executed.</li>
<li>Already running jobs keep running uninterrupted.</li>
</ul>
</td></tr>
</table>

The following table describes processing flow within a job queue for remotely executed synchronous and asynchronous jobs.

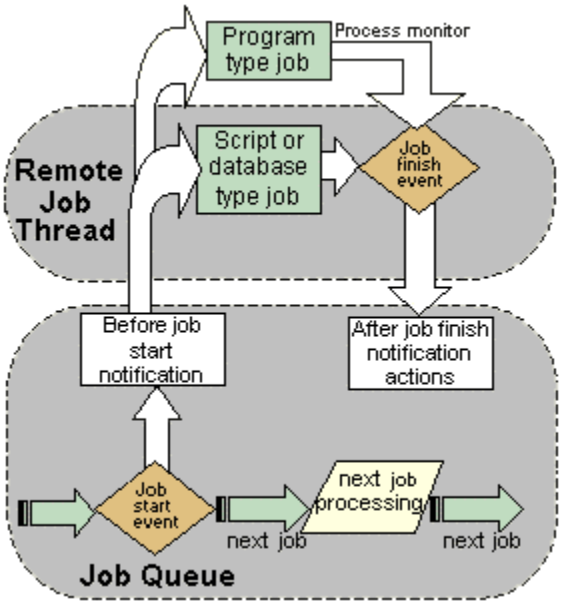
## Synchronous remote job processing




- Queued job is launched and monitored. The job queue manager waits for the job to complete before processing next queued job.
- New jobs **can** be added to the queue while this job is running.
- New jobs **cannot** be started in the same queue while this job is running. However, new jobs **can** be started in other queues while this job is running
- New jobs **cannot** be also start while the notification actions (if any) are being executed.
- Already running jobs keep running uninterrupted.

This processing type is similar to processing synchronous local jobs with the only difference that the job is executed on the remote computer.

## Asynchronous remote job processing




- Queued job is launched and monitored.
- New jobs **can** be added to the queue while this job is running.
- New jobs **can** be start from the same job queue while this job is running.
- New jobs **cannot** be started while the notification actions (if any) are being executed.
- *On error* and *on finish* notification actions can be **delayed** if the Queue Manager is busy executing other job at the time of asynchronous remote job finish. These actions are executed whenever the queue manager is free to process them.
- Already running jobs keep running uninterrupted.

**See also:**

Job Queues
Job Dependencies Editor
Job Execution Properties
Notification Events and Actions
Semaphore Files

# Job Properties
# General Properties

**Job ID**

The Job ID provides a unique descriptor for each job definition in the job database. The 24x7 Scheduler automatically generates a new job ID for each new job added to the database. The 24x7 Scheduler uses that ID to refer to the job in the job event log records.

**Job Name**

The Job Name provides a descriptive title for the job. For example, “Production Server Backup.” A unique name is not required for each job, however we highly recommend that you use unique job names. For complete job description and additional comments, you can use the **Description** property.

**Description**

This is the place where you enter job description, comments, and instructions. For your convenience, Job Explorer shows a few lines of the job description when you rest the mouse pointer over the job. Double-click the job in Job Explorer to start the Job Wizard, which you can use to see and modify the full text if necessary.

**Status**

This property indicates job status. The status can be one of the following:

- **Active** - the job is enabled and waiting for the specified job start conditions to be met
- **Disabled** - the job is disabled

**See also:**

Job Execution Properties

Job Schedule and Triggers

Notification Events and Actions

Semaphore Files

Macro-parameters

Remote Jobs

# Job Execution Properties

**Job Type**

This is the type of scheduled job. The type can be one of the following:

- **Program** – used to execute a program or a document file
- **Database command** – used to execute SQL script
- **Script** – used to execute JAL or VB script

**Program Name**

This is 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 path or filename includes spaces you must use double quotes as a module name delimiters. For example: "C:\Program Files\Daily Jobs\DataLoad.bat."

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 listed in the PATH environment variable.

You should always include the full path. Do not rely on the PATH environment variable. The PATH value may be different at the time of the program run, depending on which account the program is being run under. Remember that other programs can modify the PATH environment variable as well.

The program name can be followed by **command line parameters**.

You can use **macro-parameters** and **system environment variables** within **program name** and command line parameters string to pass dynamic information (such as the current month, name of the current user, name of the system directory, etc…) on to the scheduled program.

**Start In**

The Start In directory (e.g. *working directory*) is the directory where the program executable finds the associated files it needs to run the program. Most programs do not require an entry in this property. Occasionally however, a program will not 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.

You can use **macro-parameters** and **system environment variables** within the **Start in** string to insert some dynamic information (such as the current month). For example, this option can be used to start a program that every month searches for input files in a different directory.

**Run As Another User**

This set of properties can be used to specify the user whose credentials and security settings should be used for the job. The process created by the job runs in the security context of the specified user. If no user name is specified the job runs in the security context of the user currently logged to the system or in the security context of the LocalSystem account if not user is logged on.

**Caution:**

- **Run As Another User** option is not supported on Windows 95/98/Me systems.
- Windows NT (NT/2000/XP/2003/VISTA/2008/7) considers the user as logged on until the process (and all child processes) have ended. If the logon is successful, the system notifies network providers that the logon occurred.
- The process created in the security context of another user is non-interactive, that is, it runs on a desktop that is not visible and cannot receive user input. Also, the process inherits the environment of the 24x7 Scheduler, rather than the environment associated with the specified user.
- The user account specified in the **Run As Another User** properties must have the necessary **LOG ON AS A BATCH JOB** privilege (LOGON32_LOGON_BATCH). This user right is defined in the Default Domain Controller Group Policy object (GPO) and in the local security policy of workstations and servers. By default, only the LocalSystem account has the privilege to be logged on as a batch job.
- On Windows NT/2000 systems the user account under which the 24x7 is running must have **Trusted Computer Base** privilege (SE_TCB_NAME). The account should also have **File Change Notify** privilege (SE_CHANGE_NOTIFY_NAME). This privilege is not required for the LocalSystem account or accounts that are members of the administrators group. By default, SE_CHANGE_NOTIFY_NAME is enabled for all users, but some administrators may disable it for everyone.
- The **Trusted Computer Base** and **File Change Notify** privileges are no longer required in Windows XP for running processes in the security context of another user.

**Window Type**

This property controls type of a window to use to start the scheduled program (or document) - maximized, minimized, normal, or hidden.

If a **hidden** window type is chosen, the program, once started, will be invisible and as a result, will not be able to interact with the Desktop. One of the reasons why you may want this facility is to prevent other users from seeing or interacting with the program. However, this can lead to problems of its own. Because the program is running invisibly, you will not be able to interact with it if you do need to intervene. You should therefore, only use this option for programs that are designed to proven run without user interaction and then terminate automatically.

**Caution:** Some programs are designed the way they change window size regardless of what is the initial window state the program is started in.

**Priority**

This indicates job priority.

For program and document file jobs:

The priority property affects both priority of the job in the job queue and the priority of the started operation system process that runs the specified program.

For script and SQL database jobs:

The priority property affects both priority of the job in the job queue and the priority of the started job thread within 24x7 Scheduler instance. It does not change the priority of the 24x7 Scheduler process itself, nether it changes priority of the created database process. However, if an external program is started from a job script, 24x7 sets priority of the created operation system process that runs the specified program to the level specified by the job priority.

**Caution:** Changing the priority of a job/process can make it run faster or slower (depending on whether you raise or lower the priority), but it can also adversely affect the performance of other processes.

**Queue**

This specifies job queue, where the job will be submitted for execution. For more information about job queues see **About Job Queues** topic.

**Host** (Remote Agent)

This specifies the name of the remote agent running on the host computer that will run the job. You can use **Backup Host** property to specify one or more alternative hosts that can be used in case if this main host is not available to run the job. If this property is empty or you choose **Local** value from the drop-down list, the job will run on the same computer. If you choose **[Least busy]** value from the drop-down list, the job will run on one of the computers specified in the **Host List**property. 24x7 Scheduler will automatically determine and which host (Remote Agent) is least busy and execute the job on that computer.

**Backup Host** (Backup Remote Agent)

The name of the remote agent running on the host computer that will run the job in the event of the main host being unavailable. You can specify multiple hosts for this property using a comma-separated list. The 24x7 Scheduler always attempts to submit the job to the main host first and if the host is not available, it attempts to submit the job to the first available Backup Host. Backup Hosts are tried in the order they appear in this property.

**Host List** (Remote Agents that can be used for load balancing)

The names of remote agents running on host computer that can potentially run the job. You must specify multiple names for this property. Separate multiple names by comma. 24x7 Scheduler submits the job to the host that is least busy. 24x7 uses this property only if the **Host** property is specified as **[Least busy]**.

**Restart Windows**

This specifies whether you want to restart Windows after the job is complete. When this option is specified 24x7 logs off all active network connections, shuts down and restarts the system.
During a shutdown or log-off operation, applications (if any are still running) that are going to be shut down, are allowed a specific amount of time to respond to the shutdown request. If the time expires, Windows normally displays a dialog box that allows the user to forcibly shut down the application, retry the shutdown, or cancel the shutdown request. The 24x7 Scheduler always forces applications to close and does not display the dialog box.

**Asynchronous Process**

For program and document file jobs:

The 24x7 Scheduler executes the specified program asynchronously in background using a separate job thread, and keeps monitoring the running program (process) until it is complete or the time-out interval elapses. In the last case, 24x7 forcedly terminates the process. See also **Timeout** property. While the job is running other jobs can be started in the same job queue. For more information see **Job Processing Flow** topic. See also property.

New process inherits the Environment block from 24x7.

In order to run a document, its extension must be registered. For example: if you want to start MDB files that have the *AutoExec* macro, you must have the. MDB file extension registered as a MS Access database application.

For script and SQL database jobs:

The 24x7 Scheduler executes an asynchronous job in background using a separate job thread, taking advantage of Windows multi-tasking features. While an asynchronous job is running in background 24x7 Scheduler can run other asynchronous and synchronous jobs. 24x7 Scheduler executes a synchronous job using main queue thread (e.g. in foreground), so that other jobs cannot to start in the same queue until current job processing is complete.

**Detached Job**

This property indicates whether the scheduled must be run by a separate instance of the job engine. Enabling this property is the same as running job manually from the command line using **24x7 /JOB** command. When this option is enabled, 24x7 Schedule spawns another instance that runs the job. Depending on the job value of **Asynchronous** property, the main instance either wait or don't wait for the job to complete.
24x7 takes advantage of Windows multitasking features that allow multiple instances of the same program to run concurrently.

New process inherits the Environment block from 24x7 main instance.

**Caution:** You should enable Detached Process property for jobs that affect or can potentially affect 24x7 Scheduler stability, that is, if the job crashes or cause some memory leak, it does no affect the main instance that will continue to run other jobs even after the detached job has crashed. Note that detached processes while running take some additional system resources.

**Exit code condition**

The exit code is the value returned by the scheduled program upon termination. The **Exit Code Condition** specifies exit codes designating that the job was a failure and this will lead to 24x7 Scheduler to treat it as a failed job. Upon failure, 24x7 Scheduler may be directed to rerun the job, disable the job or execute various "on error" notification actions as specified by the job properties. For details see **Retry After Error**, **Disable On Error**, **Ignore ErrorsNotification Events and Actions**.

If the exit code is not important, leave the exit code condition property empty or [any].

This parameter is only valid for “program" type jobs.

**Timeout**

This is the time interval, in minutes that the job is allowed to run before being terminated. When the timeout occurs, 24x7 Scheduler forcibly terminates the started process. If zero is specified, the interval never occurs, therefore the process can run on to infinity. This property is used with synchronous program type jobs only, which means that 24x7 Scheduler will continue with other job processing only after this job finishes. This property makes sense for synchronous processes (program executions) only.

**Send Keystroke**

24x7 Scheduler allows you to define and send keystrokes to the program run by the job. Using this property you are able to simulate those keystrokes that you might need to further automate this program. The **Send Keystroke** option determines whether to send the keystroke to the program or the document.

**Caution:** Send Keystroke option can be used only with interactive programs running in Normal or Maximized windows only. That's it. This option cannot be used with non-interactive programs or invisible programs.

**Keystroke**
These are the actual keystrokes that are sent to the program or document.

**Init. Timeout**
This is the time that the 24x7 Scheduler will wait before sending a keystroke to the program or document it has started.

**Delay**
This is the maximum allowed delay for a late job. If the actual delay is greater than this parameter, then 24x7 Scheduler will skip a late run.

**Caution:** When scheduling daily tasks, make sure that the value of start time + maximum delay falls on the same day as the start time. If this condition is not satisfied a late job may be skipped because of the day shift.

**Skip Job if Delay is Over <n> minutes**
This instructs the 24x7 Scheduler to skip the late job where the delay is longer than the specified **Delay** interval. You use the **Delay** property to specify the maximum allowed delay (see above).

**Script**
This is the text of script that 24x7 Scheduler executes. The script programming language is defined by the Script Type property. Do not confuse this property with the script for notification actions and SQL script for database jobs.

This parameter applies to the “Script” job type only.

**Script Type**
The **Script** can be written using either **Job Automation Language** (JAL) or **Visual Basic Script** (VBS). Do not confuse this property with the script type for notification actions.

This parameter applies to the “Script” job type only.

**Profile**
The database profile name that refers to the database profile describing connection parameters for the database job. The specified profile must exist at the time that the job is being run. This parameter applies to the “Database" job type only.

**SQL**
This is the text of SQL script that the 24x7 Scheduler executes. This parameter applies to the “Database” job type only.

**Logging Enabled**
This instructs the 24x7 Scheduler whether it should log job execution statuses.

**Ignore Errors**
This instructs the 24x7 Scheduler to continue job execution regardless of job run-time errors.
Examples:
A JAL script will continue to run even if the "division by zero" error occurred.
A "job finished" notification action will be attempted even if the database job itself failed while executing SQL script.
A semaphore file assigned to the job will be deleted (using "delete after run" rule) even if the launched program terminated abnormally.

**Caution:** This option disables all other "Error" handling options as the job is never considered as failed.

**Disable On Error**

This instructs the 24x7 Scheduler to disable the job if it fails. Automatic job disabling on error ensures that the job does not restart until you find and fix the cause of the error and manually re-enable the job.

**Caution:** **Disable On Error** and **Retry After Error** options are **NOT** mutually exclusive! If the retry option is enabled, the job is not considered as failed after the first failed run. However, the **Ignore Errors** and **Disable On Error** options are mutually exclusive, because if the Ignore Errors option is enable, the job is never considered as failed.

**Retry After Error**

This instructs the 24x7 Scheduler to retry the job if it fails. The job will be restarted after the specified **Retry Interval** You should use the **Number of Retries** property to specify how many times you want the 24x7 Scheduler to retry a failing job before it marks that job as unsuccessful. If the job continues to fail, it will be restarted again and again until it is successful or the maximum number of retry attempts has been reached.

**Retry Interval**

This parameter is the number of seconds that the 24x7 Scheduler will wait before restarting the failed job. This parameter makes sense only if the job has the **Retry After Error** property enabled.

**Caution:** For jobs with time-based schedules, the total retry interval should not exceed the time interval between regular job runs. For jobs having file, process, or e-mail watch schedules, the total retry interval should not exceed the **polling interval**. To calculate the total retry interval, multiply the **Number of Retries** by the sum of the **Retry Interval** and the estimated job run time.

**Number of Retries**

This is the maximum number of attempts the 24x7 Scheduler makes to run the job. The job is considered as failed only after all attempts fail. This parameter makes sense only if the job has the **Retry After Error** property enabled.

**See also:**

General Properties

Job Schedule and Triggers

Notification Events and Actions

Semaphore Files

Macro-parameters

Remote Jobs

# Job Schedule and Triggers

A job schedule can be based on a condition that must be met before the job is executed. The 24x7 Scheduler supports six predefined types of conditions:

- **Time watch** – a time condition tests for a specified time.
- **File watch** – a file condition tests for the presence of a specified file or group of files.
- **Process watch** – a process condition tests for the presence or absence of other programs running at the same time.
- **E-mail watch** – an e-mail condition tests for the presence of a specified e-mail message.
- **User watch** – a user condition tests for a user activity (you can control the user activity threshold via the settings on the currently selected Screen Saver; click the button  to open the display settings dialog. This trigger is not supported on Windows 95).
- **System watch** – a system condition tests for a user log off or system shutdown

**User-defined conditions:** You can also check your own conditions using **Script** type jobs.

**Tip:**
Use **macro-parameters** to build flexible job triggers. For example, to define a file-watch trigger for a file whose name is different every day, you can use @T macro-parameter within file name.
Example: f:\data\@T"mmddyy".txt file name can be used for a job that should run whenever a new file having 011501.txt, 042201.txt, or other name in "month-day-year" format appears in f:\data directory.

The following parameters are used to specify job conditions:

**Repeat Every**
This is the interval at which 24x7 Scheduler will rerun the job.

**Start Date**
This is the first date you want the job to run. This property is used in combination with the **Start Time**. The 24x7 Scheduler will not allow an invalid date such as 1/32/95. The required format is **mm/dd/yy**. The 24x7 Scheduler uses the following rule to convert a 2-digit year to internal date format: If the number is equal to or less than 50 then it falls into the 21st Century, otherwise it belongs to the 20th Century. You can also use the up and down arrows to increase/decrease the days, months, and years.

**Start Time**
This is the first time you want the job to run. This property is used in combination with the **Start Date**. The required format is **hh:mm**. You must use the 24-hour time format. . While on this control you can use the up and down arrows to increase/decrease the hours and minutes.

**End Date**
This is the last date after which you do not want the job to run. This property is used in combination with the **End Time** parameter. The 24x7 Scheduler will not allow an invalid date such as 1/32/95. The required format is **mm/dd/yy**. The 24x7 Scheduler uses the following rule to convert a 2-digit year to internal date format: If the number is equal to or less then 50 then it falls into the 21st Century, otherwise it belongs to the 20th Century. You can also use the up and down arrows to increment/decrement the days, month, and years.

**End Time**
This is last time after which you do not want the job to run. This property is used in combination with the **End Date**. The required format is **hh:mm**. You must use the 24-hour time format. You can also use the up and down arrows to increase/decrease the hours and minutes.

**All Day Schedule – Daily Start Time**
This property can be used to specify run-time limits for jobs having **All Day** schedule. This property is used in combination with the **Daily End Date**. To specify Daily Start and End Time click the **Advanced** button. The Specify Day-time parameters dialog will appear. The required format for the time properties is **hh:mm:ss**. You must use 24-hour time format. While on this control you can use arrows up and down to increment / decrement the hours, minutes and seconds. By default Daily Start Time is set to 00:00:00.

**All Day Schedule – Daily End Time**
This property can be used to specify to run-time limits for jobs having **All Day** schedule. This property is used in combination with the **Daily Start Date**. To specify Daily Start and End Time click the **Advanced** button. The Specify Day-time parameters dialog will appear. The required format for the time properties is **hh:mm:ss**. You must use 24-hour time format. While on this control you can use arrows up and down to increment / decrement the hours, minutes and seconds. By default Daily End Time is set to 23:59:59.

**[K]All Day Schedule – Fixed Time List**
This property can be used to specify fixed times for a job having **All Day** schedule. If the fixed time list is specified the schedule is not recursive during day-time interval. The job starts only at the specified times. Time values in the list must be specified in either **hh:mm** or **hh:mm:ss** 24-hour time format.

[K] All Day Schedule, Fixed Time List

Example: 9:00,9:20,10:00,10:20,11:00,11:20

**Caution:** If you want a job to run just once a day, use **Daily or Weekly** schedule type instead of **All Day** schedule type because **Daily or Weekly** schedule type is better optimized for such processing.

**Daily and Weekly Schedule – Days of Week**

These are the days of the week when the job is scheduled to run. Select the desired days of week by placing checkmarks in the appropriate boxes.

**Monthly Schedule – Days of Month**

This is the day of the month when the job is scheduled to run. Choose the desired day of month by day number (for example, the 2nd day of every month) or by day name (for example, 1st Monday of every month) or by day type (for example, last weekday of every month).

**Monthly Schedule – Fixed Day List**

These are the fixed days when the job is scheduled to run. You can enter multiple days as a comma-separated list (for example, 1,3,7,15.)

**Skip Job On Holiday**

This property indicates whether the 24x7 Scheduler must skip the job on the scheduled day if that day is a holiday.

**Slide Job On Holiday**

This property indicates whether the 24x7 Scheduler must skip the job on the scheduled day if that day is a holiday and then run the job on the next non-holiday.

**Semaphore Files**

This is the name of the semaphore file that the job will “watch" i.e. file condition. A file condition tests for the presence of the specified file or group of files. The job starts after the 24x7 Scheduler finds the file. You may specify more that one file. Use comma to separate multiple semaphore files. For example, you could specify the following file names securities.dat, holdings.dat, accounts.dat. You may also use standard wildcard characters in file names. For example: c:\data*.rbc, c:\data\an??b.rb?.

**Semaphore Process**

This is the name of the main process module that the job will “watch" i.e. process condition. A process condition tests for the presence or absence of another running program or process. The job starts after the 24x7 Scheduler finds the process (if the criteria is set to check for presence of the process) or after it does not find the process (if the criteria is set to check for absence of the process). For example, if you want to take some actions when your Oracle database is not running, you would enter oracle80.exe as the process name. Sometimes this job type is called “server watch".

**Polling Interval**

Use this property to specify how often you want the 24x7 Scheduler to check for the specified file, process, or e-mail message. You can specify this parameter in minutes See **Semaphore Files** and **Semaphore Process** parameters described above.

**Caution:** The 24x7 Scheduler can check for a file or group of files every second. If you specify short polling intervals you can improve file detection accuracy. However, frequent checking may cause high CPU utilization and heavy network traffic when verifying network files. Use balanced approach when choosing appropriate polling intervals.

**Monitor Semaphore File Size Stability**

This property allows you to specify whether you want the 24x7 Scheduler to monitor sizes of found semaphore files and how often do you want check and compare file size changes. Use this property in jobs that process the monitored semaphore files. This can help to avoid situations when the job starts but the process that created

them is still writing the files. To disable the checking and have the job to start as soon as the specified files are found, leave the Check Interval field blank or enter zero.

**Caution:** Too short check interval can lead to premature job start.

**Delete Semaphore File Rule**

This property allows you to specify post-execution or pre-execution action for a “file watch” job. Three actions are supported:

- Delete the semaphore file (or group of files) before actual job run.
- Delete the semaphore file (or group of files) after actual job run.
- Do not delete the semaphore file (or group of files). The 24x7 assumes that the job itself takes care of the semaphore files. If the job leaves those files alone, the 24x7 Scheduler starts the same job next time it checks for the files again. The file check frequency is specified by **Polling Interval** parameter (see above).

This parameter can be used only in “file watch” jobs.

**Account**

This is the e-mail account name that the 24x7 Scheduler will use to check an e-mail message. This parameter is only valid for the “e-mail watch” jobs. The value for the e-mail account may be different for different e-mail interfaces. For the MAPI interface you should specify the name of the MAPI profile that you use when logging-into the e-mail system. For Lotus Notes you should specify the name of the user (or ID) that you use when logging-in to the Lotus Notes.

**Password**

This is the e-mail password that the 24x7 Scheduler will use to check for e-mail messages. This parameter is only valid for “e-mail watch" jobs.

**Message Text**

This is the e-mail message text that the 24x7 Scheduler uses for comparison when checking an e-mail message. It is only valid for “e-mail watch" jobs only.

**Subject**

This is the e-mail message subject that the 24x7 Scheduler uses for comparison when checking an e-mail message. It is only valid for “e-mail watch" jobs only.

**Save Attachments**

This instructs the 24x7 Scheduler to save attachments found in the e-mail message that triggered the job. This parameter is only valid for “e-mail watch" jobs. 24x7 Scheduler saves attachments in the user's temporary directory. Text of the email message is saved in *message.txt* file.

**See also:**

General Properties

Job Execution Properties

Notification Events and Actions

Semaphore Files

Macro-parameters

Remote Jobs

# Notification Events and Actions

**Job Notification Options**

A job may issue a notification at a given event such as job start, job completion or job execution error.

This option may be used to:

- Notify system administrators and operators of events. This is especially convenient for controlling critical processes and may be used to page operations personnel in the event of a job failure. This enables operations personnel to respond rapidly to and correct any production issues.
- Create conditional e-mail reminders.
- Create semaphore files used for job interdependencies so that depended jobs can start as soon as they detect this semaphore file(s). This allows linking multiple jobs in one logical batch process as well as invoking jobs on demand such as “fix it" jobs. The 24x7 Scheduler allows very flexible job interdependencies. It even allows you to link jobs on multiple local and remote computers provided the 24x7 Remote Agents or 24x7 Masters are running on the remote computers.
- Log job events to centralized corporate database.
- Execute notification scripts for advanced job notification processing.
- Send SNMP traps to enterprise network monitoring and management console such as HP Open View, Microsoft MMC and other.
- Run external programs to perform advanced correction actions.
- Trigger other jobs for perform advanced correction actions and also to perform advanced job notification processing.

**Notification Events**

The following notification events are supported:

- **Start** – occurs just before a job is started.
- **Finish** – occurs immediately after a job is successfully finished.
- **Not found** – occurs when program file or document file specified in job properties cannot be found leading to the situation when a job cannot start. This event occurs after the **Start** event.
- **Error** – occurs when a job fails.
- **Late job** – occurs if a job was started later than it was scheduled to start. This event occurs immediately after the **Start** event. Note that if the **Skip Late Job** option is enabled, the job is never started if it is late and no **Late Job** event is generated. The Maximum **Delay** job parameter defines what is a late job.

 **Tips:**

- You can specify more than one event to be processed. You can also specify different notification actions for different events of the same job.
- You can use JAL or VBS script notification types to create customized event handlers that will give you greater control and allow different notification processing for different events. In the script you can use the @V"event" **macro-parameter** to find out which event indeed triggered the script.

**Notification Actions**

The following actions can be performed on the notification events:

- **Send e-mail –** Sends an e-mail message to the specified e-mail address. The message subject is fixed - “Message from 24x7 Scheduler". The message text depends on the triggered notification event and may be one of the following:
  1 “Job <name> started.”
  2 “Job <name> finished.”
  3 “Job <name> execution error: <error description>.”
  4 “Job <name> execution error: Program not found.”
  5 “Job <name> started late. Delay is <xxx> minutes.”

  The <name> parameter is substituted with the actual job name that the job was given. The <error description> parameter is substituted with the actual error text. If an error occurs and logging is enabled, the error is automatically written in the log file. For more information on error types and the error log, see Job Activity and System Events Logs topic.

  For this action type you must specify:
  1 E-mail profile (account) to use for sending the notification message. For example: *Exchange Settings* The actual value may be different for different e-mail interfaces. For MAPI interface you should use name of the MAPI profile that you use when logging-on to the e-mail system. For Lotus Notes you should use the name of the user (or ID) used when logging-on to Lotus Notes. For SMTP interface you should use your e-mail address.
  2 E-mail password.

3 E-mail recipient. Use comma (,) to separate names of recipients.

- **Send pager message –** Sends a pager message to the specified pager. To send pager messages 24x7 uses standard SNPP protocol (RFC 1861). The message text depends on the triggered notification event and may be one of the following:
  6 "Job <name> started."
  7 "Job <name> finished."
  8 "Job <name> execution error: <error description>."
  9 "Job <name> execution error: Program not found."
  10 "Job <name> started late. Delay is <xxx> minutes."

  The <name> parameter is substituted with the actual job name that the job was given. The <error description> parameter is substituted with the actual error text. If an error occurs and logging is enabled, the error is automatically written in the log file. For more information on error types and the error log, see Job Activity and System Events Logs topic.

  For this action type you must specify message recipient's pager number. The number format depends on your carrier. Please contact your carrier to find out the proper format. You can also visit www.snpp.info web site for more information about SNPP. Use comma (,) to separate names of recipients.

- **Send network message –** Sends a network message to the network user or computer. This notification action is not supported on Windows 95/98/Me. The message text depends on the triggered notification event and may be one of the following:
  11 "Job <name> started."
  12 "Job <name> finished."
  13 "Job <name> execution error: <error description>."
  14 "Job <name> execution error: Program not found."
  15 "Job <name> started late. Delay is <xxx> minutes."

  The <name> parameter is substituted with the actual job name that the job was given. The <error description> parameter is substituted with the actual error text. If an error occurs and logging is enabled, the error is automatically written in the log file. For more information on error types and the error log, see Job Activity and System Events Logs topic.

  For this action type you must specify message recipient's name in the **name/domain** format (for example 'ADMIN/LANDMN. Use comma (,) to separate multiple recipients.

  Note, the effect of this action is similar to the execution of the Windows NT *NET SEND* command from the DOS prompt.

- **Execute SQL command(s)** – Executes SQL commands using the specified database profile.
  For this action type you must specify:
  1 Name of the database profile to be used when executing the notification message.
  2 SQL script to send to the back-end database.

  In the SQL script you can specify any valid SQL statement(s) supported by your database. You can use also available @V macro-parameters to insert job/event context information. For example the following SQL can be used to insert data into corporate log database:

```
INSERT INTO my_log_table ( event_time, computer, job, job_name, event_type, message )
VALUES ( GetDate(), '@V"computer"', @V"job_id", '"@V"job_name"', '@V"event"', 'Example
message');
```

- **Create semaphore file(s) –** Creates semaphore files. The file contents depend on the triggered notification event and can be one of the following:
  1 "Job <name> started."
  2 "Job <name> finished."
  3 "Job <name> execution error: <error description>."
  4 "Job <name> execution error: Program not found."
  5 "Job <name> started late. Delay is <xxx> minutes."

  The <name> parameter is substituted with the actual job name that the job was given. The <error description> parameter is substituted with the actual error text. If an error occurs and logging is enabled, the error is automatically written in the log file. For more information on error types and error log, see Job Activity and System Events Logs topic.

For this action type you must specify:
1 Full name of the file to be created, including path to the file. Use comma (,) to separate multiple files.

- **Execute script** – Executes specified script.
  For this action type you must specify:
  1 Script type. The **Script** can be written using either **Job Automation Language** (JAL) or **Visual Basic Script** (VBS). Do not confuse this property with the script type for the job itself.
  2 Script to be executed in the notification action. Do not confuse this property with the script for job itself and SQL script for Database Command notification action.

- **Send SNMP Trap –** Uses local SNMP agent to send a SNMP Trap message to the network management console. This notification action is not supported on Windows 95/98/Me. The trap contains two variable bindings: Job ID that generated the trap and Job Status. The JOB ID variable binding has INTEGER data type; the Job Status variable binding has STRING data type. The Job Status value depends on the triggered notification event and may be one of the following:
  16 "Job <name> started."
  17 "Job <name> finished."
  18 "Job <name> execution error: <error description>."
  19 "Job <name> execution error: Program not found."
  20 "Job <name> started late. Delay is <xxx> minutes."

  The <name> parameter is substituted with the actual job name that the job was given. The <error description> parameter is substituted with the actual error text. If an error occurs and logging is enabled, the error is automatically written in the log file. For more information on error types and the error log, see Job Activity and System Events Logs topic.

  For this action type to work properly the SNMP service must be running and 24x7 SNMP Extension Agent must be installed on the system. To configure the SNMP Service, use Control Panel's Services applet. Select the SNMP Service and go to 'Properties'. Configure the 'Agent', 'Traps' and 'Security' tab settings.

- **Run external program –** Runs an external operation system command or program.
  For this action type you must specify:
  1 Program name. The **Program Name** may contain full or partial program name that you want to start. The program name may be following by command line parameters. If the program name includes spaces enclose it in double quotes, for example, *"c:\Program Files\program.exe" parm1 parm2*.
  2 Timeout - This is the time interval, in minutes that the process is allowed to run before being terminated. When the timeout occurs, 24x7 Scheduler forcibly terminates the started process. If zero is specified, the interval never occurs, therefore the process can run on to infinity.

- **Run job** – Runs an external operation system command or program.
  For this action type you must specify:
  1 Job ID or Job Name. The **Job ID** or **Job Name** of the job that you want to run**.**

**See also:**

General Properties
Job Execution Properties
Job Schedule and Triggers
Semaphore Files
Macro-parameters
Remote Jobs

# Macro-parameters

Macro-parameters provide a way to replace specific values at run time. Macro-parameters can be used in the following areas:

- Semaphore file names

- Program names and directories
- SQL commands
- Notification messages
- Job automation scripts

The @ sign is used as an escape character for macro-parameters. If you need a literal @ sign (for example, in a program name specification) you must double it, for example, “@@somename”.

**Parameter Substitution**
The 24x7 Scheduler always applies Macro-parameters before checking the job conditions and executing the job. There are two parameter types: time-dependent parameters and general-purpose V parameters that simplify job design.
Some macro-parameters consist of two parts: The parameter identifier and the format mask. You must use double quotes around masks in order for parameter to be interpreted correctly. See the following tables for supported parameters and masks.

24x7 Scheduler supports two types of macro-parameters:
- Time-dependent parameters
- V-parameters

**Time-dependent parameters**

The following time-dependent macro-parameters are supported in all places:

| Identifier | Meaning |
|---|---|
| @W | An integer (1-7) representing the current day of the week. Sunday is day 1, Monday is day 2, and so on. |
| @T" <mask>" | Current date and time. The actual representation depends on the mask. |
| @Q | An integer (1-4) representing the current quarter of the year. |
| @D<number>"<mask>" | The date of a specified day number of the current week. The actual representation depends on the mask. The number can be anything from 1 to 7, where Sunday is day 1, Monday is day 2, and so on. |
| @DP" <mask>" | The date for the previous calendar day. The actual representation depends on the mask. |
| @DN" <mask>" | The date for the next calendar day. The actual representation depends on the mask. |
| @DL" <mask>" | The date for the last business day. The actual representation depends on the mask. |
| @DB" <mask>" | The date for the next business day. The actual representation depends on the mask. |
| @ML" <mask>" | The date for the last calendar day of the month. The actual representation depends on the mask. |
| @MF" <mask>" | The date of the first calendar day of the month. The actual representation depends on the mask. |
| @ME" <mask>" | The date for the last (ending) business day of the month. The actual representation depends on the mask. |
| @MS" <mask>" | The date for the first (starting) business day of the month. The actual representation depends on the mask. |

**Format Masks**

| Character | Meaning | Example |
|---|---|---|
| d | Day number with no leading zero. | 9 |
| dd | Day number with leading zero, if appropriate. | 09 |
| ddd | Day name abbreviation. | Mon |
| dddd | Day name. | Monday |
| m | Month number with no leading zero. | 6 |
| mm | Month number with leading zero, if appropriate. | 06 |
| mmm | Month name abbreviation. | Jun |
| mmmm | Month name. | June |
| yy | Two-digit year. | 97 |
| yyyy | Four-digit year. | 1997 |

Colons, slashes, and spaces appear as they are specified in the mask.

**Two-digit years**

If you specify a two-digit year in a mask, where the year is greater than or equal to 50, the 24x7 Scheduler will assume that the date is in the 20th Century, If the year is less than 50, the 24x7 Scheduler will assume it is the 21st Century. For example:

1/1/85 is interpreted as January 1, 1985. 1/1/40 is interpreted as January 1, 2040.

**Examples:**

The following table shows how the date Friday, Jan. 30, 1998, displays when different format masks are applied:

| Format | Displays |
|---|---|
| mmddyy | 013098 |
| mmyyyyy | 011998 |
| d-mmm-yy | 30-Jan-98 |
| dd-mmmm | 30-January |
| mmm-yy | Jan-98 |
| dddd, mmm d, yyyy | Friday, Jan 30, 1998 |
| dddd | Friday |

**V-parameters**

The **V** macro-parameters are similar to other macro-parameters described above. The scope of each macro-parameter is specified in the parameter description. V-parameters are case-sensitive. You can use the following **V** macro-parameters in your jobs:

| Identifier | Meaning |
|---|---|
| @V"subject" | This parameter can be used in scripts for email-watch jobs. It returns full subject of the email message that triggered the job. |
| @V"x-subject" | This parameter can be used in scripts for email-watch jobs. It returns full subject of the email message that triggered the job formatting escape symbols and double quotes, essentially transforming the text to a text value that can be |

| | |
|---|---|
| | inserted directly into a job script and not cause syntax errors. |
| @V"message" | This parameter can be used in scripts for email-watch jobs. It returns full text of the email message that triggered the job. |
| @V"x-message" | This parameter can be used in scripts for email-watch jobs. It returns full text of the email message that triggered the job with all carriage return and new line characters replaced with spaces. It also formats escape symbols and double quotes, and essentially transforms the text to a single line value that can be inserted directly into a job script and not cause syntax errors typically caused by regular @V"message" macro-parameter. |
| @V"datereceived" | This parameter can be used in scripts for email-watch jobs. It returns receiving date-time of the email message that triggered the job. The value is retuned as a string in the format it appears in the message header. |
| @V"sender" | This parameter can be used in scripts for email-watch jobs. It returns sender ‘s email address from the email message that triggered the job. |
| @V"attachments" | This parameter can be used in scripts for email-watch jobs. It returns comma-separated list of files attached to the email message that triggered the job. If there is no file attached, it returns empty string |
| @V"user" | Returns user id of the user currently logged on. This parameter can be used in any script, notification event or job trigger. |
| @V"computer" | Returns name of computer on which the job is running. This parameter can be used in any script, notification event or job trigger. |
| @V"domain" | Returns name of the primary logon domain for the user currently logged on. This parameter can be used in any script, notification event or job trigger. This macro-parameter is not supported on Windows 95/98/Me. |
| @V"job_id" | Returns id of the job whose script is referencing this macro-parameter. This parameter can be used in any script, notification event or job trigger. |
| @V"job_name" | Returns name of the job whose script is referencing this macro-parameter. This parameter can be used in any script, notification event or job trigger. |
| @V"24x7_home" | Returns name of the 24x7 Scheduler installation directory. This parameter can be used in any script, notification event or job trigger. |
| @V"files" | This parameter can be used in scripts and notification events of jobs that have "file-watch" triggers. This macro-parameter returns comma-separated list of files that triggered the job. |
| @V"event" | This parameter can be used in scripts and properties for job notification actions. Depending on the notification event that triggered the action, the returned value can be one of the following:<br>• START<br>• FINISH<br>• ERROR<br>• NOT FOUND<br>• LATE |
| @V"winver" | Returns Windows version and service pack number. This parameter can be used in any script, notification event or job trigger. |
| @V"param:[N]" | Returns value of the command line parameter passed to the command line JAL script. [N] represents relative position of the parameter specified on the command line. First parameter specified after JAL file name is counted as 1. See the following examples for more details. |
| @V"env:[VAR]" | Returns value of the DOS/Windows environment variable whose name is referenced by [VAR]. This parameter can be used in any script, notification event or job trigger. |

**Examples:**

The following statement coded in “Macro-testing” job will display “Hello from job Macro-testing” message
MessageBox “Hello from job @V”job_name””

The following line will add user-defined message to the job log. Note that the job id and name are not hard-coded and in run-time they are be substituted with the actual job id and name.
LogAddMessageEx “INFO”, @V”job_id”, “@V”job_name””, “Some message here”

The following code fragment will delete disk versions of email attachments
```
FileDeleteEx "@V"attachments""
```

The following code fragment will send automatic email confirmation to the email message sender.
```
MailSend "Exchange Settings", "mypassword", "@V"sender"", "RE: @V"subject"", "Thank
you. Your message has been received.\r\n\r\n Truly yours, @V"user""
```

The following code fragment will assign default Java class path (as specified in the %CLASSPATH % environment variable) to the script variable v_classpath.
```
Set v_classpath, "@V"env:CLASSPATH""
```

The following code fragment will assign value of the first command line parameter "test parameter" specified in the following command to the to the script variable v_param1. The last line will assign value of the second command line parameter "test parameter 2" specified in the following command to the to the script variable v_param2

*JALScript C:\scripts\some_script.jal "test parameter" "test parameter 2"*

```
Set v_param1, "@V"param:1""
Set v_param2, "@V"param:2""
```

# System Environment Variables

System environment variables provide a way to replace specific values at run time. Environment variables can be used in the following areas:

- Program names and directories in job properties
- **Run**, **RunAndWait**, **RunAsUser**, **RunAsUserAndWait**, **RunConfig**, **RunWithInput** statements within JAL script jobs.

The percent symbol (%) is used to delimit names of environment variables, just as it can be used in DOS batch files.

**Parameter Substitution**

The 24x7 Scheduler always substitutes environment variables before checking the job conditions and executing the job. If the scheduler cannot find the specified environment variables it removes it from the command line. For complete list of available environment variables that can be used on your system run SET command from the DOS prompt.

Example command line:
```
%WINDIR%\notepad.exe "%USERPROFILE%\My Documents\mytext.txt"
```

# Adding New Job

**To schedule a new task**

1 In the Job Explorer, select the folder in which you want to create a new job.
2 Select **File** menu, then select **New**, and then choose **Job** (shortcut CTRL N). Alternatively you can click the toolbar button 

3 Follow the instructions in the Job Wizard.

4 You must save changes in the job database in order to apply them to the active job pool.

**Notes**

- Make sure that the system date and time for your computer are accurate. The 24x7 Job Scheduler relies on this information to know when to run scheduled tasks. To check or change the date and time, double-click the time on the status bar.

**See also:**

Working with Job Database

# Deleting Job

**To delete a scheduled job**

1 In the Job Explorer, select job that you want to delete.

2 Select **File** menu then select **Delete** command. Alternatively you can click the toolbar button 


3 You must save changes in the job database in order to apply them to the active job pool.

**Tips:**

- You can also right-click on the job in the Job Explorer then select **Delete** command from the context menu.
- The delete function removes job definition from the job database. Any program files associated with that job are not removed from the hard disk.
- If you deleted a job by mistake, you can quit 24x7 Scheduler without saving changes. Then restart 24x7 Scheduler and it will reload the job database from the disk file.

**See also:**

Stop Running Job

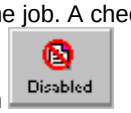
Working with Job Database

# Disabling/Enabling Job

A job state is either disabled or enabled. You can toggle the job state by setting the on/off state of the **Disabled** property.

**To temporarily disable a scheduled job**

1 In the Job Explorer, select job you want to disable.

2 Select **File** menu, then select **Disabled** command. The 24x7 Scheduler will disable the job. A checkmark will appear next to the **Disabled** menu item. Alternatively, you can click the toolbar button 


3 You must save changes to the job database in order to apply them to the active job pool.

**To enable a previously disabled job**

Repeat the steps described above.

**Tips:**

- You can also right-click on the job in Job Explorer, then select the **Disabled** command from the context menu.
- Disabling scheduled job does not remove job definition from the job database; it only removes the job from the active job pool.

**See also:**

Stop Running Job

Working with Job Database

# Modifying Job Definition and Schedule

**To rename a job**

1 Right-click on a job in the Job Explorer.
2 Select **Rename** command
3 Type new job title; press **Enter** key.

**To disable/enable a job**

See Disabling/Enabling Job topic above for detailed instructions.

**To change all other job properties**

You can use the Job Wizard to modify most job properties. To run **the Job Wizard** double-click the job in the Job Explorer or highlight the job then press the F4 key.

# Moving Job to Another Folder

Use drag-and-drop to perform this operation. See Drag and Drop Interface topic for details.

**See also:**

Working with Job Database

# Copying Job to/from Another Database

You can use the Job Database Manager tool to copy jobs between two different job databases. Select Tools/Job Database Manager menu item to start the Job Database Manager.

For more information see **Job Database Manager** topic

**See also:**

Working with Job Database

Moving Job to Another Folder

# Protecting/Unprotecting Job

24x7 Scheduler provides several ways of restricting access to a job. You can do any of the following:

- Assign a password to modify the job. This prevents unauthorized users from modifying, deleting or disabling the job, but allows them to view job properties and scripts as well as execute the job.

- Assign a password to view and modify the job. This allows other users to execute the job but not to modify, delete or disable/enable the job.
- Assign a password to execute, view, and modify the job. This does not allow other users to execute, modify, delete or disable/enable the job.

**Caution:** If you assign password protection to a job and then forget the password, you can't remove protection from it, or recover data from it. It's a good idea to keep a list of your passwords and their corresponding job names in a safe place.

**To protect / unprotect a job:**

1. Select the desired job in the **Job Explorer**.
2. Select **File/Protect Job** command. The job protection dialog box will appear.
3. Type your password and select the desired protection level.
4. Click the **OK** button.

**Notes:**

- "Execute" protection does not prevent the job from starting on schedule or from being started by another job. It simply does not allow unauthorized users to execute the job by clicking on the **Run Now** button or by clicking the **File/Run Now** menu item. You can prevent unauthorized access to the 24x7 Scheduler by using existing Windows security mechanisms and policies.
- Jobs copied using the **Database Manager** tool retain their protection features.

**See also:**

Working with Job Database

# Testing Job Execution

You can test a job without waiting for the scheduled event to happen. One of the following options can be used to start the job immediately:

- Right-click the job you want to start, and then click **Run Now**.
- Click on job you want to start, and then select the **File/Run Now** command from the menu (shortcut CTRL R), alternatively you can click on the toolbar button


**Important note:**
When an error occurs in the **Run Now** mode, 24x7 Scheduler will display modal message boxes that require input from the user. You should always close these messages as they can, potentially, prevent other jobs from starting in time.
Normally, the 24x7 Scheduler displays non-modal message boxes only. These non-modal message boxes will close by themselves after a few minutes. This allows 24x7 Scheduler to work unattended. You can also close these messages manually by clicking on the desired message box button.
You can use system options to set display times for a non-modal message box (click **Tools** menu, click **Options** menu, then specify the desired display time in minutes in the **Maximum error message box display time** field).

**Tips**:

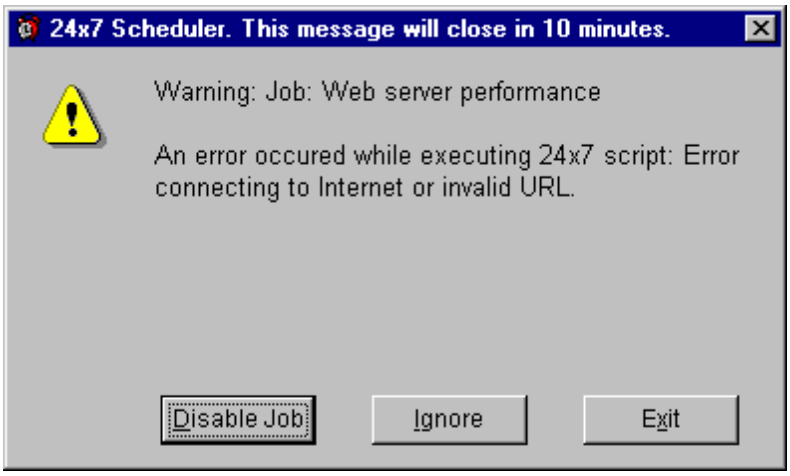
- You can stop the job using Windows Task Manager. Launch Windows Task Manager by pressing once CTRL+ALT+DELETE (If using Window NT or 2000 or XP), then click on **Task List** button to start the Task Manager), then select the task and click the **End Process** button.
- If a task should have run but did not, check the log to see why. Select the job in the Job Explorer, then click the **Log** tab.

**See also:**

Disabling/Enabling Job

# Error Messages

When an error occurs in the **Run Now** mode, the 24x7 Scheduler will display modal message boxes that require input from the user. You should always close these messages as they can, potentially, prevent other jobs from starting in time.
Normally, the 24x7 Schedule displays non-modal message boxes only. These non-modal message boxes will close by themselves after a few minutes. This allows 24x7 Scheduler to work unattended. You can also close a non-modal message manually by clicking on the desired message box button. See error message box example below.




Button description:

**Disable Job** - disable job that caused the error and close the message box.

**Ignore** - Ignore error and close the message box.

**Exit** - Shutdown the 24x7 Scheduler.

**Tip:**
You can use system options to set the display time for a non-modal message box. Click **Tools** menu, click **Options** menu, then specify the desired display time in minutes in the **Maximum error message box display time** field.

# Stopping Job Execution

**To stop a scheduled job that is currently running**

You can stop a particular job by using Window Task Manager. To launch Windows Task Manager press once CTRL+ALT+DELETE (if using Window NT or 2000 or XP, click on the **Task List** button to start the Task Manager), then select the task and click the **End Process** button.

**Note:**

- If a scheduled program starts another program, the stop procedure described above stops only the scheduled program, not the secondary program. You should repeat the same procedure for the secondary program.

# Creating Job Shortcut

**Executing a single job from the DOS command line**

You can start a single job by executing the following command from the DOS system prompt:
24x7 /JOB <job id>

This command starts 24x7 Scheduler. The scheduler, in turn, starts your job, waits for the job to complete, and then terminates as soon as the job is finished.

**Executing a single job from the Windows Start Menu**

You can create a Windows shortcut to every a job by using the 24x7 Scheduler with the following parameters: /JOB <job id>

This shortcut can be placed anywhere in the Windows Start or Programs menu so that a job can be started by selecting the appropriate menu item. If you place a shortcut to the **Startup** folder, it will be executed every time Windows starts. You can also place a shortcut to the Windows Desktop. This means that you can start your favorite jobs simply by double-clicking on their Desktop icons.

**Notes:**

- While running a single job specified in the command line parameters, the 24x7 Scheduler does not process other events and jobs. The specified job is always processed synchronously.
- If a program being executed in the single job mode starts another program, the second program may keep running even after both the first program and the 24x7 Scheduler terminate.
- If a script being executed in the single job mode starts a program asynchronously by using the Run statement, the started program may keep running even after both the script and the 24x7 Scheduler terminate.

# Job Templates

# Using Job Templates

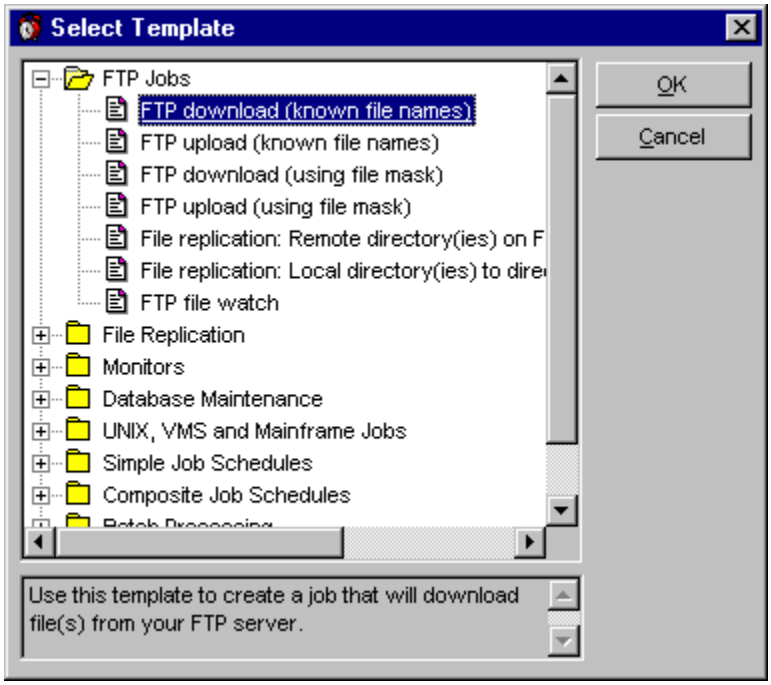
Job templates provide a quick way to create new jobs. Some templates are installed by default with 24x7 Scheduler Setup, other are created by users. Additional templates are also available on SoftTree Technologies, Inc. web site. If you have access to the Internet, click the **Help** menu in 24x7 Scheduler graphical interface, and then click **Tips and Scripts Archive**. Follow instructions on the Web page to download additional templates.

Most templates can be used to create ready-to-run jobs. The rest can be used to automate most of the job creation work, while requiring from the user to complete the remaining job properties.

**To schedule a new task from a template**

1 In the Job Explorer, select the folder in which you want to create a new job.
2 Select **File** menu then select **New** command, and then choose **Job** (shortcut CTRL N). Alternatively you can click the toolbar button 

3 Click on the **Create from template** link. The **Select Template** dialog box will appear.




4 Select the desired template, and then click **OK** button. The **Template Wizard** will appear. Follow the instructions in the **Template Wizard**. When finished click the **OK** button. The Template Wizard will setup a new job based on your instructions. If necessary complete other job properties
5 Follow the instructions in the **Job Wizard**.
6 You must save changes in the job database in order to apply them to the active job pool.

**Notes**

- Make sure that the system date and time for your computer are accurate. The 24x7 Job Scheduler relies on this information to know when to run scheduled tasks. To check or change the date and time, double-click the time on the **status bar**.

**See also:**

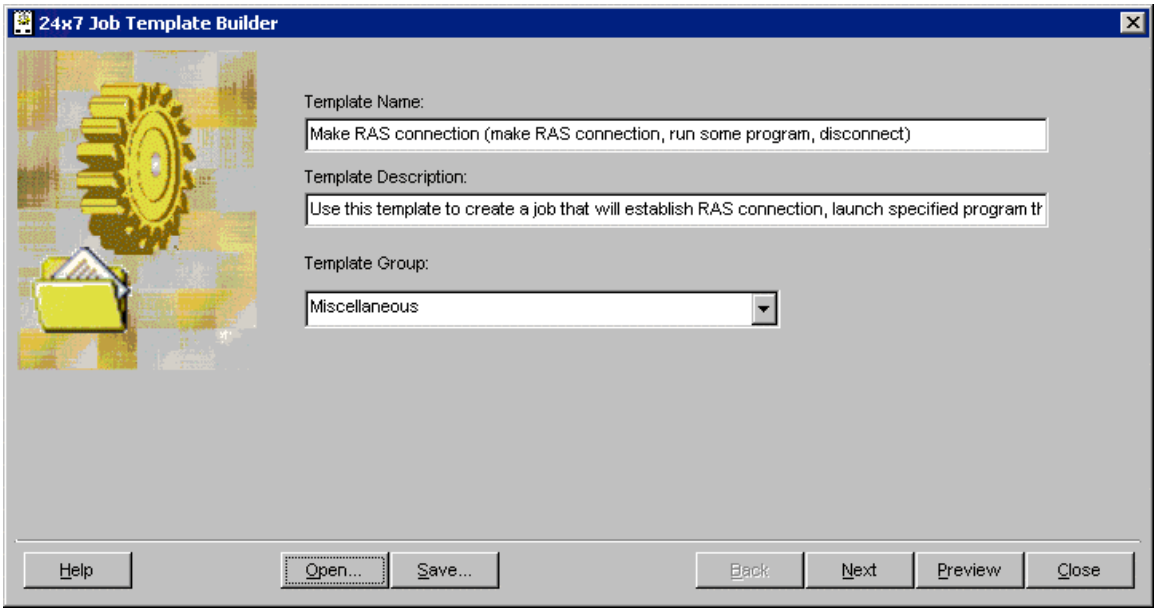
Working with job database

Job Properties Wizard

Creating New Template

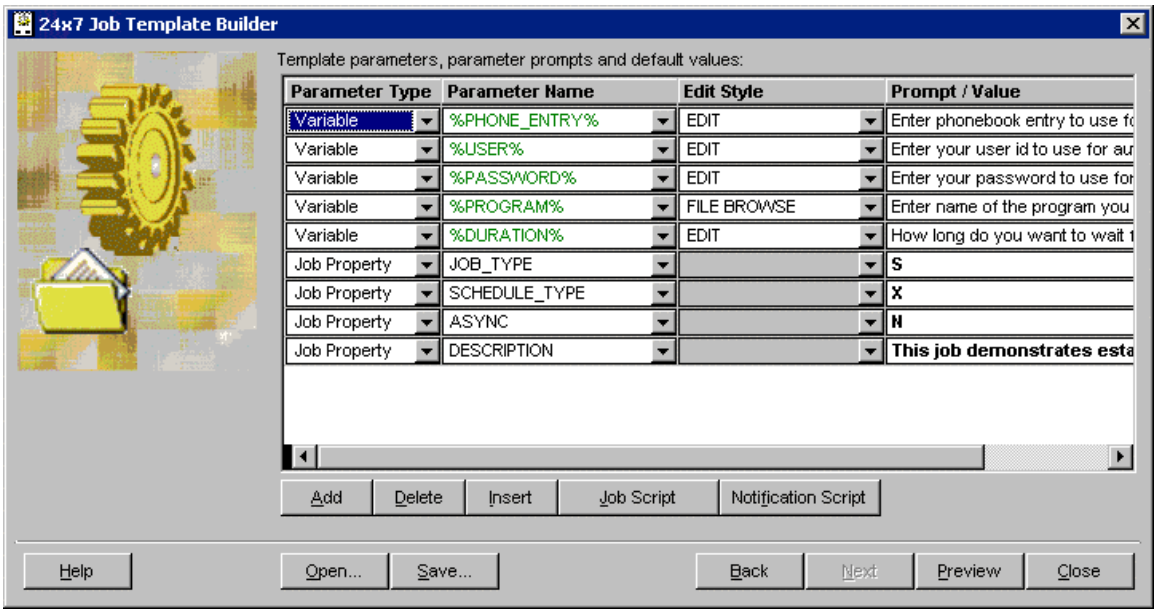
# Creating and Modifying Job Templates

You can use the Job Template Builder tool to create new and modify existing job templates. To start Job Template Builder click **Tools/Job Template Builder** menu.




**To create a new template**

1 Start Job Template Builder.
2 Enter name of the new template and optionally enter template description.
3 Choose the appropriate template group from the Template Group drop-down box or type in the new group name. Click the Next button.
4 Enter desired job properties and template variables. See **Job Property Names** help topic for description and names of available job properties.
5 Click the Save… button to save new template. The Save File dialog will appear. Select the name of the file in which new template will be saved and then click the OK button.




**To create a new template from an existing template**

1 Start Job Template Builder.
2 Click the Open button to select and load an existing template. The Select template dialog will appear. Select the desired template and click the OK button.
3 Modify template name and optionally enter new template description.

4 Choose the appropriate template group from the Template Group drop-down box or type in the new group name. Click the Next button.
5 Enter desired job properties and template variables. See **Job Property Names** help topic for description and names of available job properties.
6 Click the Save… button to save new template. The Save File dialog will appear. Select the name of the file in which new template will be saved and then click the OK button. **Make sure to enter the template new file name, do not override an existing template file**

**To modify an existing template**

1 Start Job Template Builder.
2 Click the Open button to select and load an existing template. The Select template dialog will appear. Select the desired template and click the OK button.
3 If required modify template description. **Do not modify template name and group**. Click the Next button.
4 Modify desired job properties and template variables. See **Job Property Names** help topic for description and names of available job properties.
5 Click the Save… button to save new template. The Save File dialog will appear. If desired choose a new template file or simply click the OK button. If a new template file is chosen, the old file will not be deleted.

**Tips:**

- Variable names must be enclosed in % signs. When applying the template for a new job, 24x7 Scheduler replaces variable names appearing in the template definition with the values entered in the Prompt/Value column.
- Names of variables can be used for values of variable job properties. When entering such values into the Prompt/Value column, make sure the specified variable is already defined above. Enter the name exactly as it appears in the variable definition including the % signs.
- For complete description of edit styles see **InputBox** topic.
- If the template is designed for a script type job use the Job Script button to enter or edit the job script. The script can include names of variables already defined in the template. When applying the template for a new job, 24x7 Scheduler replaces variable names appearing in the script with the values entered in the Prompt/Value column.
- To specify script type define SCRIPT_TYPE parameter with one of the following values: SQL, VBS, or JAL
- If the template contains definitions of notification actions of script type use the Notification Script button to enter or edit the notification action script. The script can include names of variables already defined in the template. When applying the template for a new job, 24x7 Scheduler replaces variable names appearing in the script with the values entered in the Prompt/Value column.
- To specify script type for notification action define NOTIFY_SCRIPT_TYPE parameter with one of the following values: SQL, VBS, or JAL
- You cannot specify different notification scripts for different notification events, as there could be only one notification script created for a job. Instead program some logic the script that performs different operations based on the notification event type. You can use the **@V macro-parameter** to find out event that triggered the notification action.
- It is a good idea to study templates provided with 24x7 Scheduler to find out how you can elegantly build your own job templates using the most advanced features of the 24x7 Scheduler.

**Notes:**

- Job Templates are not stored in the job database. They are stored in separate .INI files, which are located by default in the Template subdirectory. The template names and group definitions are stored in the TEMPLATE.INI file. You can find this file in 24x7 Automation Suite installation directory. When backing up or copying 24x7 jobs database and configuration files to another computer do not forget to copy templates that you want to use on that computer.
- If your templates contain sensitive information like passwords use Operation System security features to limit access to these template files. It is also a good idea not to hard-code such sensitive information and design your templates such that the sensitive information must be entered by the user in response to the **Template Wizard** prompt.

**See also:**

Job Property Names
Job Properties Wizard
Using Job Templates

# Sending Keystrokes To Other Programs

You can define a series of keystrokes, which can be sent automatically as part of the job execution process. Only use the **Send Keystroke** functions to operate other applications when there is no alternative, and even then use it with caution. You should test sending a keystroke under a variety of conditions to avoid unpredictable results, data loss, or both.

The following table lists non-character keys, which can be sent:

| **Key** | **Code** |
|---|---|
| Tab | {TAB} |
| Enter | {ENTER} or {CR} or {RETURN} |
| Esc | {ESCAPE} or {ESC} |
| Backspace | {BACKSPACE} or {BS} or {BKSP} |
| Break | {BREAK} |
| Caps Lock | {SCROLLLOCK} or {SCROLL LOCK} |
| Caps Lock | {CAPSLOCK} or {CAPS LOCK} or {CAP} |
| Del | {DELETE} or {DEL} |
| Down Arrow | {DOWN} |
| End | {END} |
| Help | {HELP} |
| Home | {HOME} |
| Ins | {INSERT} |
| Left Arrow | {LEFT} |
| Num Lock | {NUMLOCK} or {NUM LOCK} |
| Page Down | {PGDN} or {PAGE DOWN} or {PAGEDOWN} |
| Page Up | {PGUP} or {PAGE UP} or {PAGEUP} |
| Print Screen | {PRTSC} or {PRINT} or {PRINT SCREEN} |
| Right Arrow | {RIGHT} |
| Up Arrow | {UP} |
| F1...F12 | {F1}...{F12} |
| Ctrl | {CTRL} |
| Shift | {SHIFT} |

| | |
|---|---|
| Alt | {ALT} |
| Numpad 0...Numpad 9 | {NUMPAD0}...{NUMPAD9} or {NUMPAD 0}...{NUMPAD 9} |
| Numpad + | {NUMPAD+} or {NUMPAD +} |
| Numpad - | {NUMPAD-} or {NUMPAD -} |
| Numpad * | {NUMPAD*} or {NUMPAD *} |
| Numpad / | {NUMPAD/} or {NUMPAD /} |
| Numpad . | {NUMPAD.} or {NUMPAD .} |
| Numpad Enter | {NUMPADENTER} or {NUMPAD ENTER} or {NUMPAD RETURN} or {NUMPAD CR} |

To send a key combination that includes SHIFT, ALT, or CTRL, use the following methods:

- When sending a special key with a single character, precede the character code with one of the special keys: {CTRL} for Ctrl, {SHIFT} for Shift and {ALT} for Alt. For example: to send the key combination **Alt M** use **{ALT}m**
- When sending a special key with a group of characters, use parentheses to group the characters and precede the group code with one of the special keys: {CTRL} for Ctrl, {SHIFT} for Shift and {ALT} for Alt. For example: to send the key combination **Alt M** then **Alt G** then **Alt X** use **{ALT}(mgx)**
- To send braces and parenthesis **{, }, (, )** as literal text, enclose these character in braces. For example, to send an open parenthesis use **{(}**.
- To pause between keystrokes use **{WAIT n}**, where **n** is the duration in seconds. For example, to send **A** pause for 10 seconds, then send **B** then pause for another 5 seconds and then send **C** specify **a{WAIT 10}b{WAIT 5}c**.

**Note:**

- When sending key combinations that include the ALT key, make sure to send them in lowercase characters. For example, to open the File menu **ALT F**, use **{ALT}f**. Using **{ALT}F** is the same as pressing **ALT+SHIFT+F** that might trigger other action.

**Example:**

123 A. {SHIFT}(:"):"{ENTER}{ALT}{F4}N
The keystroke above is equivalent to pressing keys 1, 2, 3, space, A, period, :, ", ;, ', Enter, Alt+F4, N

# Job Queues

## About Job Queues

The 24x7 Scheduler supports multiple concurrent job queues. Each queue has its own Queue Processor that instantly scans the queue for new submitted jobs and executes them according to their **priority**, order in queue, and processing type. High priority jobs are executed first, then go jobs with normal priority, and then go low priority jobs. Jobs having same priority are executed in the order they were submitted to the queue, but before jobs having less priority. The Queue Processor executes jobs one-by-one as they become available in the queue. However, job processing maybe different depending on the job **asynchronous property** state. The Queue Processor executes synchronous jobs synchronously, meaning that other pending jobs in the same queue wait for the Queue Processor to complete the current job before moving to the next one. The asynchronous jobs are spawned and executed by a separate processing thread created for every asynchronous job so that the Queue Processor and other jobs do not have to wait for the current job to complete.

On the other side, the 24x7 Job Event Processor instantly scans jobs in the active job pool and checks their start conditions (i.e. schedule and events). The Job Event Processor evaluates these conditions and submits jobs whose start conditions are satisfied to the appropriate job queue. The Job Event Processor and the Queue Processors operate independently so that the Job Event Processor can submit a job even while the Queue Processor is busy executing some other job.

For more information about job processing see Job Processing Flow topic.

The 24x7 Scheduler offers a Queue Monitor GUI, which gives the user a detailed insight into the Queue Processor's internal state. It should be stressed that the Queue Monitor, can be connected to any active queue and it operates in a real-time. The Queue Monitor can be also used to manage already queued jobs.

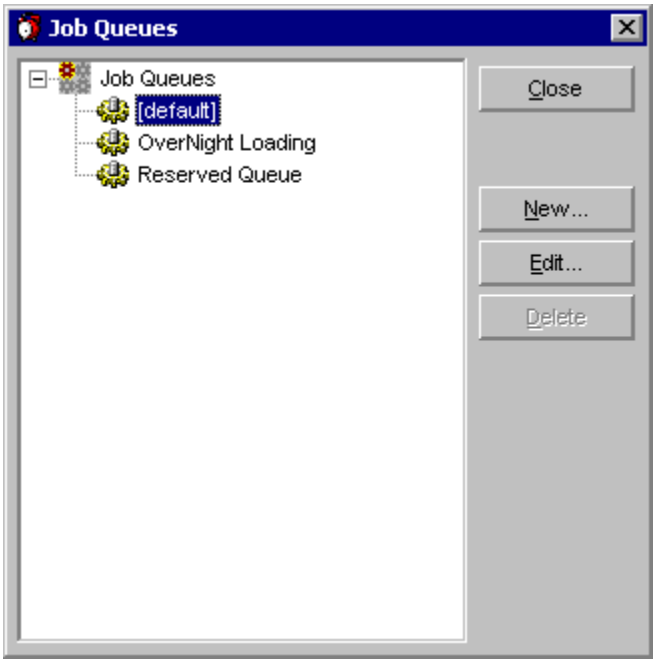
**Notes:**
There are several important features that you should know and carefully consider before setting the 24x7 job queues:

- Every configured job queue requires additional memory for a dedicated copy of the job engine. Every asynchronous job also requires additional memory for a dedicated copy of the job engine so that the job can run in its own virtual environment.
- Running asynchronous jobs is a resource "expensive" function. Whenever possible try to avoid asynchronous jobs. It is recommended that you use asynchronous jobs only for "non-stop" processing. For example, a job with a script implementing logical loop "forever", is a good candidate for an asynchronous job.
- If you don't want a job to block other jobs, setup it to run in another job queue.
  Or if want a job to be always executed immediately without taking chances that it will be waiting sometime for the Queue Processor to become available, setup it to run in a dedicated job queue.
- 24x7 Remote Agents do NOT use job queues. They always execute submitted jobs immediately. However, if the number of jobs to be processed at same time is greater than the number of configured job threads, every additional job will wait for a free Agent job thread before it can be executed. The default number of job threads on the Agent is 16. If this number is not sufficient you can reconfigure it.
- Not all jobs are queued for execution. Jobs triggered in the "system" events such as startup, shutdown, and user "idle" are always executed immediately and **out-of-queue**.
- **When you exit 24x7 Scheduler, all running jobs (if any) are forcedly terminated.**

**See also:**
Job Processing Flow
Creating and Modifying Queues
Monitoring and Managing Queued Jobs

# Creating and Modifying Queues

The 24x7 Scheduler supports multiple concurrent job queues. You can create as many queues as anticipated. Just keep in mind that every additional queue requires about 4MB of virtual memory.
The [default] queue is required and cannot be deleted.




To add, modify, and delete queues, click the **Tools** menu, and then click **Job Queues**. The queue profiles dialog will appear.

**To add a new queue**

1 Click the **New** button.
2 Enter Queue name and disk storage parameters.
3 Click the **OK** button.

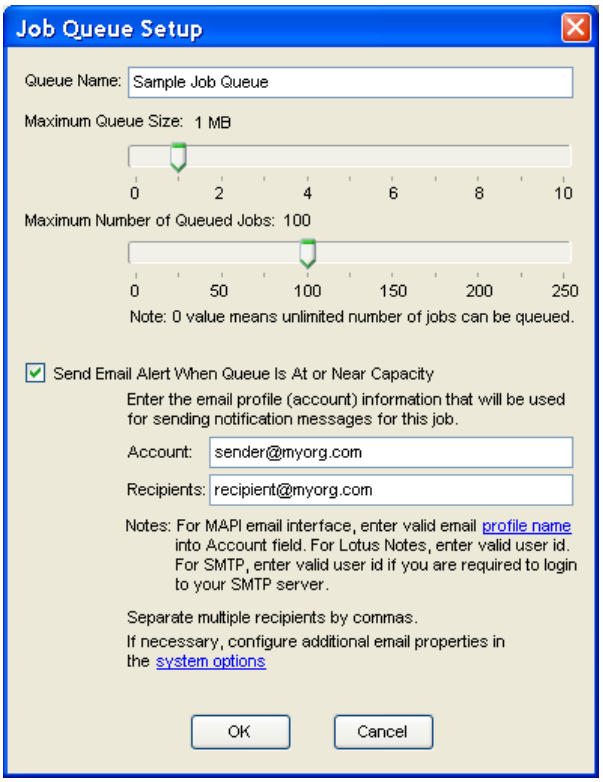
**To modify an existing queue**

1 Select the desired queue in the **queue** list (see queue profiles dialog above).
2 Click the **Edit** button.
3 Modify queue information.
4 Click the **OK** button.

**To delete a queue**

1 Select the desired queue in the **queue** list (see queue profiles dialog above).
2 Click the **Delete** button.

### Job Queue Properties




- Every queue must have a unique name.
- Every queue has user-defined disk space usage limit. This limit is used to prevent a situation when jobs pile up in a job queue faster than the system can process them. In case a queue reaches the specified size, new jobs will not be added immediately and will wait for a free space in that queue to become available. The default space limit is 1MB that normally sufficient to fit many hundreds of jobs. When a job is processed, the job definition is removed from the queue and the released space can be reused for another job. You can adjust job queue size to meet your unique requirements.
- Maximum number of jobs allowed in a queue provides another method to control runaway jobs. If this value is set to zero, the scheduler does not check for the number of running jobs.
- Send email alerts option enables email alerts. The scheduler sends such alerts in case a queue is at or near its maximum capacity. By default, the scheduler checks and alerts on queue space utilization only. In case the maximum number of jobs parameter is set to a non-zero value, the scheduler also checks for the number of running and queued jobs.
- If you have enabled email alerts for queue utilization, make sure to specify which account can be used to send them and who should be receiving such alerts. You can specify multiple individual and group email addresses in the recipient property. Note that different settings can be used for different queues.

**See also:**

Job Processing Flow
About Job Queues
Monitoring and Managing Queued Jobs

# Monitoring and Managing Queued Jobs

Jobs can pile up in a job queue without anyone knowing about it. The Job Queue Monitor utility, which is provided with the 24x7 Scheduler, will let you know if that happens.

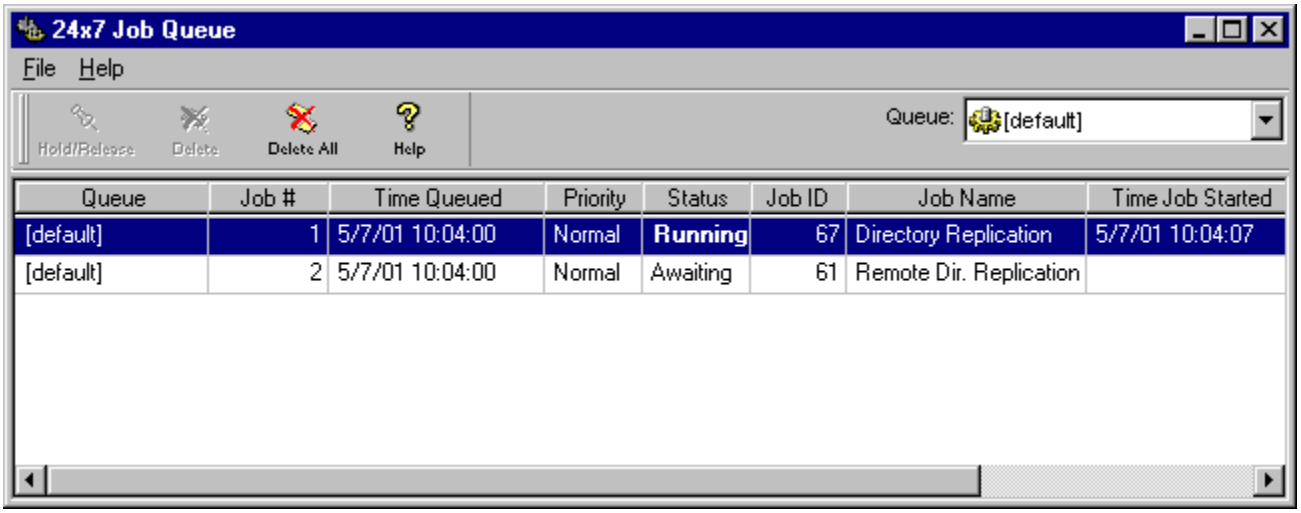
Job Queue Monitor is used to monitor and, if necessary, make modifications to jobs that have been submitted to the job queue. All-important information about scheduled jobs is stored in the job queue.

**To open the Job Queue Monitor:**

1 From the **Tools** menu, select the **Queue Monitor** item or simply click 

button on the Job Explorer window toolbar.

The Job Queue Manager appears.




Each job in the queue is listed separately with detailed information. You can view/modify the following fields:

Queue – Name of the queue the Queue Monitor is connected to.
Job # - Job number in queue identifying uniquely identifying job instance in the queue. The Queue Processor assigns this number to each job submitted to the queue.
Time Queued – This is the date and time when the job was added to the queue
Priority – The job priority.
Status - A job will have one of the following types of status:
RUNNING - The job is currently running.
AWAITING - The job is ready to be executed and waiting for the Queue Processor to become available.
HELD - The job is held by operator or because of an error and it will not be run until it is released by operator.
Job ID – Job ID in the Job Database. You can also see that number in the job properties view in the Job Explorer.
Job Name – Job ID in the Job Database.
Time Job Started – This is the date and time when the job has been started. This applies only to jobs having RUNNING status.
Size – Size of job definition data before it is compressed in stored in the queue
Size in Q – Size of job definition data after it is compressed in stored in the queue
Ratio – Compression ratio for the job definition data

{bmct tip.bmp}**Tips:**

- You can use the Queue Monitor to change job priorities after they were added to the queue, put jobs on hold or release them, and delete all or some queued jobs that you don't want to run. **Modify priority**, **Hold/Release Job**, and **Delete** options are available under the Queue Monitor's **File** menu.
- Modifying jobs in the queue does not affect job definitions.
- Deleting a job from the queue, does not delete that job from the Job Database and that job can run again later if the job scheduling conditions will be satisfied again.
- Use the Queue drop-down list available on top of the Queue Monitor window to attach the monitor to a different queue.

**See also:**

Job Processing Flow
About Job Queues
Creating and Modifying Queues

# Troubleshooting Job Execution

If the job you schedule does not run when you expect it to, double-click the job in the Job Explorer to open the Job Wizard.

1 If the job depends on one or more semaphore files, verify that the specified files exist.
2 Make sure the job is **Enabled**.
3 Make sure the schedule is set correctly.
4 If a scheduled program does not run correctly, you may need to supply command arguments for it. To find out more about a program and its arguments, try one of the following: if the program has online help, look up that help. Try typing the following at a command prompt, where *program* is the name of the program: *program* /?
5 If a scheduled database command does not run, check the setting for the database profile. Make sure the database is available. Try connecting to the database from other programs.
6 If you are using Windows NT (NT/2000/XP/2003/VISTA/2008/7) you can gather job execution statistics (see "Collecting job execution statistics" for more details). Check them to make sure the job has enough system resources to run successfully.

# Job Execution Statistics.

When running jobs, 24x7 Scheduler is capable of capturing those jobs' execution statistics. These can be used to measure job performance as well as the amount of various system resources required to run these jobs. Not all statistics are available on Windows 95/98/Me platforms. **All statistics are available on Windows NT/2000/XP/2003/VISTA/2008/7 platforms only**. You can use these statistics to help you better understand the behavior of scheduled programs, such as processors and memory usage, run-time duration and delays. All this information is added to the STAT.LOG file after each program run. A sample entry from this file is illustrated below. Note that some memory load statistics may not be completely accurate as the system memory is shared among all the processes and Windows NT (NT/2000/XP/2003/VISTA/2008/7), as a multi-tasking system, is capable of running multiple parallel tasks simultaneously. However, analysis based on a long historical period can provide you with a good understanding of how the system is being used and the kinds of requirements needed for the scheduled programs. 24x7 Scheduler extensively analyzes the collected information in a wide variety of **Reports**.. The collected information may also help you when troubleshooting those jobs that are causing problems or anomalies.

**To enable performance statistics collection:**

1 Click **Tools** menu then select **Options** item. The System Options dialog appears.
2 Select **Log** tab page
3 Enable **Job execution statistics** option
4 Click the OK button and restart the Scheduler

**Sample entry from the statistics file**

Job No: 25 Job Name: Loan Interest Change
Process ID: 189
Creation Time: 28-Oct-1998 5:00:03:0002 Exit Time: 28-Oct-1998 5:00:33:0004
Duration: 30 seconds
Kernel Time: 0:00:00:0000 User Time: 0:00:00:0000
Exit Code: 65535

| | | |
|---|---|---|
| Free Resource: | Before: 57.784% | After: 57.384% |
| Memory Load: | Before: 0% | After: 0% |
| Available Physical Memory: | Before: 44875776 | After: 43266048 |
| Available Virtual Memory: | Before: 1981403136 | After: 1981394944 |
| Available Page File Size: | Before: 102236160 | After: 101527552 |

# Disabling Timer

In case if you have difficulties starting 24x7 Scheduler or modifying job schedules while jobs are running, you can start 24x7 Scheduler with the /NOTIMER option. To use that option, run 24x7 Scheduler from the DOS command line as 24x7 /NOTIMER. This will disable the timer used by the Event Processor and as a result it will halt all job processing until the 24x7 Scheduler is restarted without the /NOTIMER option. All other 24x7 Scheduler features and functions will continue to operate.

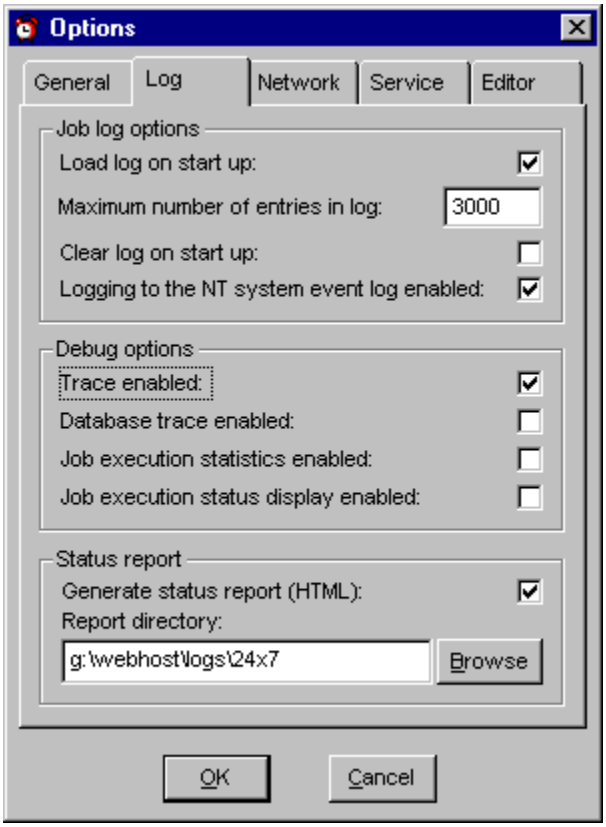
# Trace Features

The tracing features allow you to collect and analyze information about the execution of the 24x7 Scheduler. The collected information can help you identify problem areas.

When tracing is turned on, the 24x7 Scheduler records all activity in the selected area in a log file. This information is also placed in the console window. You can use the Log Viewer to analyze the complete trace log.

There are seven areas in which you can collect trace data:

- 24x7 Master/Standby Scheduler communication session
- 24x7 Remote Agent session
- Internal commands that 24x7 Scheduler performs while communicating with your database
- ODBC API calls made by 24x7 Scheduler while connected to an ODBC data source
- JAL script execution
- JDL script execution
- External interface session



To enable tracing for:

- 24x7 Master Scheduler session (on the Master Scheduler's computer)
- 24x7 Standby Scheduler session (on the Standby Scheduler's computer)
- 24x7 Remote Agent session (on Remote Agent's computer)
- JAL scripts execution tracing
- JDL script files execution tracing
- External interface session

you will need to do the following:

1 Select **Tools** menu, and then click **Options**. The system options dialog appears.
2 Click **Fail-over** tab.
3 Check **Trace enabled** option.
4 Click **OK** button.

To enable tracing for a 24x7 Remote Agent session (on the main Scheduler's computer while connected to the Remote Agent):

1 Select **Tools** menu, then click **Remote Agents**. The Remote Agent profiles dialog box appears.
2 Select the desired Agent, click **Edit**. The Agent Profile settings dialog box appears.
3 Check **Trace enabled** option.
4 Click **OK** button.
5 Click **Close** button.

To enable database tracing:
1 Select **Tools** menu, and then click **Options**. The system options dialog box appears.
2 Click **Fail-over** tab.
3 Check **Database trace enabled** option.
4 Click **OK** button.

To enable ODBC API Call tracing:
1 Select **Tools** menu, and then click **Database Profiles**. The Database Profiles dialog box appears.
2 Click **ODBC** button. The ODBC Driver Manager dialog box appears.
3 Click **Tracing** tab.
4 Specify desired tracing options, then click **OK** button.
5 Click **Close** button.

**See also:**

Database Profiles
Remote Agent Profiles
Troubleshooting Job Execution
Job Activity and System Events Logs
About Fail-over Mode

# Chapter 5: Job Interdependencies

## Overview

You may need to setup a job whose run depends on the success or failure of other scheduled jobs. In other words, a program run may be based on the outcome of other programs. The most reliable and proven mechanism is to use file dependencies. The 24x7 Scheduler provides all the necessary tools to implement this semaphore file-based linked program run.

You can have a job dependent on more than one job/file by creating multiple dependencies. To setup a dependent job, you need to choose a "file watch" condition for that job. When setting up a "parent" job, you will need to select a "notify file" option for that job.

The 24x7 Scheduler includes the easy-to-use graphical Job Dependencies Editor, which allows you to create new, or modify and delete existing dependencies with just a few mouse clicks.

**See also:**

Dependencies Editor Interface
Adding New Job to Dependencies View
Deleting Job from Dependencies View
Arranging Jobs on Dependencies View
Adding New Dependency
Deleting Dependency
Printing Dependencies

## Graphical Dependencies Editor

**To open the Dependencies Editor:**

1 From the **Tools** menu, select the **Job Dependencies Editor** item or simply click 

button on the Job Explorer window toolbar.

The Dependencies Editor appears.

The Dependencies Editor presents all available jobs as a hierarchical structure "Tree View" on the left side of the screen. This side is filled with folders and jobs. The right side of the screen displays the dependencies diagram. This side is blank unless there is at least one created dependency. The Dependencies Editor automatically recognizes all existing dependencies

To change the size of either side of the window, drag the bar that separates the two sides. Use scroll bars to navigate both sides of the Dependencies Editor window.

**"Tree View"**

If a folder has been expanded, and its contents displayed in the **Properties** view area, the folder will be represented by an open folder icon  .

Collapsed folders are represented by a closed folder icon  .

Folders with a "+"symbol next to the folder name mean that there are jobs beneath the folder.

Conversely, a "-"symbol next to a folder icon means that there are no further jobs beneath.

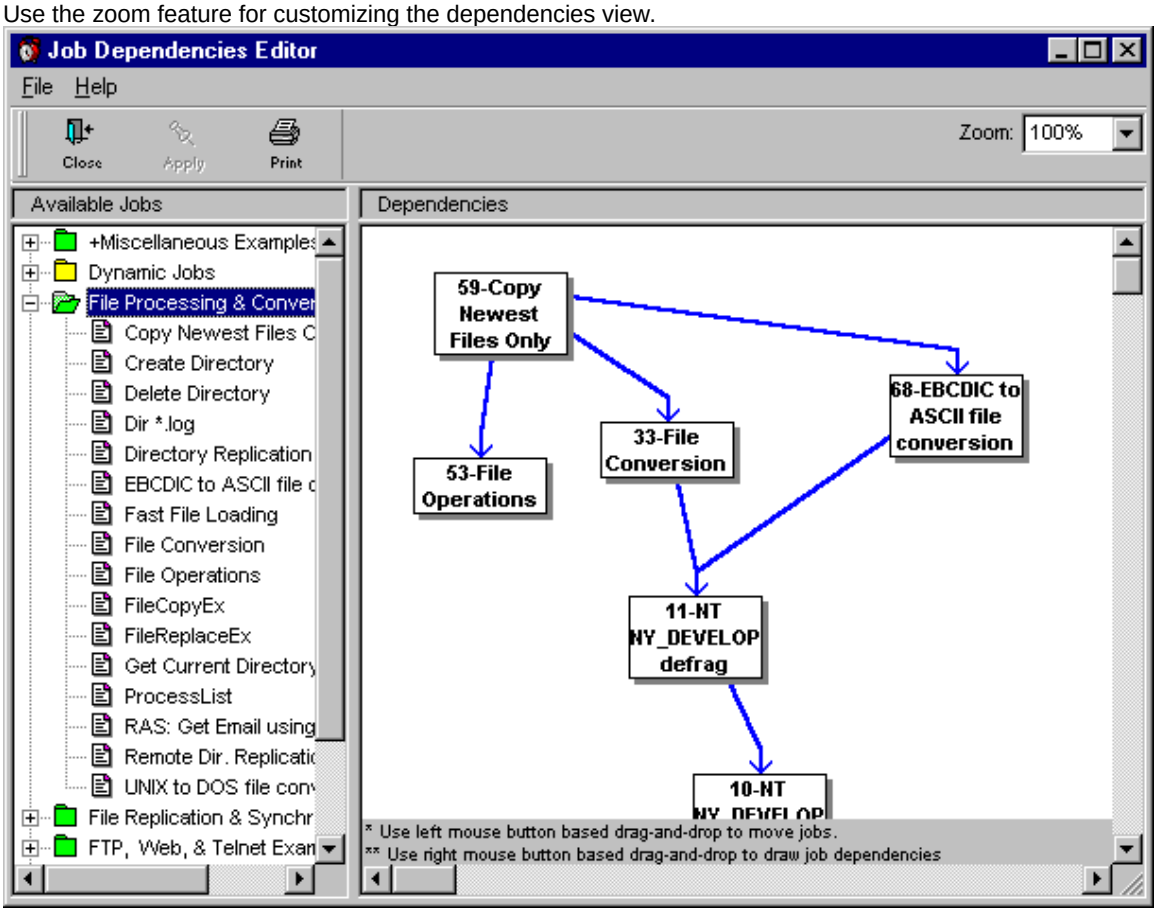
Navigating through the various folders and jobs is usually accomplished by clicking individual folders and jobs with the mouse. Click the plus sign (+) to expand a folder, or you can double-click on the folder itself. Click the minus sign (-) to collapse a folder, or you can double-click on the folder itself.

**Dependencies Diagram**

All "child" and "parent" jobs are drawn as rectangles with a number and text within it. The number is the job ID and the text, the job name.

Job dependencies are represented by a blue line ending with arrow going from the "parent" job to the dependent "child" job.

Use the zoom feature for customizing the dependencies view.




**Important:**
After you have made all the changes you want to the dependencies diagram, you should click the **Apply** button to apply them to the job database. The file image of the Job Database, however, will not be updated until you click the **Save** button on the Job Explorer toolbar. If you want to discard changes, click the **Close** button or, if you have already closed the Dependencies Editor, exit the 24x7 Scheduler without saving changes.

**Tips:**

- To view the job description, rest the mouse pointer over the job's icon on the left side of the Dependencies Editor; information about the job appears as a ToolTip under the mouse pointer.
- Avoid "dead" (circular) dependencies - a case when Job A depends on Job B and at the same time, Job B depends on Job A.

**See also:**

About Job Interdependencies
Adding New Job to Dependencies View
Deleting Job from Dependencies View
Arranging Jobs on Dependencies View
Adding New Dependency
Deleting Dependency
Printing Dependencies
Using Zoom Tool

# Adding New Job to Dependencies View

The Job Dependencies Editor supports standard drag and drop interface for moving jobs in and out of dependencies diagram.

When you drag a job icon from the Dependencies Editor tree on the left side and drop this job on the dependencies view (right side), the 24x7 Scheduler creates an object on the right side that represents this job. Alternatively, you can double-click a job icon in the Dependencies Editor tree on the left side. When you drag a folder icon and drop it on the Dependencies View, the 24x7 Scheduler creates objects on the right side that represent every job from this folder. The number of created objects matches the number of jobs in the folder.

The 24x7 Scheduler will not create a second job object If a job object already exists on the right side. Therefore, it will ignore all jobs that already have an object created.

To add a job to the Dependencies View:

1 Click on the desired job.
2 While holding the mouse button down, drag (move) the mouse pointer to the desired location on the diagram pane.
3 Release the mouse button to drop the job.

To add all jobs from one folder to the Dependencies View:

1 Click on the desired folder.
2 While holding the mouse button down, drag (move) the mouse pointer to the desired location on the Dependencies View.
3 Release the mouse button to drop the folder.

If necessary, use drag and drop on the right side to adjust object position.

**See also:**

About Job Interdependencies
Dependencies Editor Interface
Deleting Job from Dependencies View
Arranging Jobs on Dependencies View
Adding New Dependency
Deleting Dependency
Printing Dependencies
Using Zoom Tool

# Deleting Job from Dependencies View

To delete a job from Dependencies View:

1 Open Dependencies Editor.

2 Click on the **job object** rectangle.
3 Press the **Delete** button on the keyboard. A message box will appear asking you to confirm this operation.
4 Click **Yes** to confirm the deletion.

**See also:**

About Job Interdependencies
Dependencies Editor Interface
Adding New Job to Dependencies View
Arranging Jobs on Dependencies View
Adding New Dependency
Deleting Dependency
Printing Dependencies
Using Zoom Tool

# Arranging Jobs on Dependencies View

To arrange job object rectangles on the job Dependencies View:

1 Click on the desired job object rectangle.
2 While holding the mouse button down, drag (move) the mouse pointer to the desired location on the Dependencies View.
3 Release the mouse button to drop the job object. The 24x7 Scheduler will update the object position. It will also arrange all the lines that represent this object's dependencies, if it has any.

**See also:**

About Job Interdependencies
Dependencies Editor Interface
Adding New Job to Dependencies View
Deleting Job from Dependencies View
Adding New Dependency
Deleting Dependency
Printing Dependencies
Using Zoom Tool

# Adding New Dependency

To add a new job dependency:

1 Open Dependencies Editor.
2 On the right side of the Dependencies Editor window right-click on the desired "parent" job object rectangle.
3 While holding down the **right** mouse button, move the mouse pointer towards the desired "child" job rectangle.
4 Release the **right** mouse button to complete this operation. The 24x7 Scheduler will add the new dependency to the Dependencies View.

**See also:**

About Job Interdependencies
Dependencies Editor Interface
Adding New Job to Dependencies View
Deleting Job from Dependencies View
Arranging Jobs on Dependencies View
Deleting Dependency
Printing Dependencies

Using Zoom Tool

# Deleting Dependency

To delete a job dependency:

1 Open Dependencies Editor.
2 Click on the line that represents the desired dependency.
3 Press the **Delete** button on the keyboard. A message box will appear asking you to confirm this operation.
4 Click **Yes** to confirm the deletion.

**See also:**

About Job Interdependencies
Dependencies Editor Interface
Adding New Job to Dependencies View
Deleting Job from Dependencies View
Arranging Jobs on Dependencies View
Adding New Dependency
Printing Dependencies
Using Zoom Tool

# Printing Dependencies

To print the job dependencies diagram:

1 Open Dependencies Editor.
2 Click the **Print** button.

**See also:**

About Job Interdependencies
Dependencies Editor Interface
Arranging Jobs on Dependencies View
Using Zoom Tool

# Using Zoom Tool

The Zoom options on the Dependencies Editor window let you view the dependencies diagram at different magnification levels. You will be able to increase the zoom value and magnify the diagram, decrease the zoom value and reduce the size of the diagram.

 **Tip:**

- When you print a diagram, the Zoom option is not ignored. The diagram will be printed at the specified magnification. To print the diagram at normal magnification, change Zoom to 100%.

**See also:**

About Job Interdependencies
Dependencies Editor Interface
Arranging Jobs on Dependencies View
Printing Dependencies

Using Zoom Tool

# Chapter 6: Fail-over Mode

## About Fail-over Mode

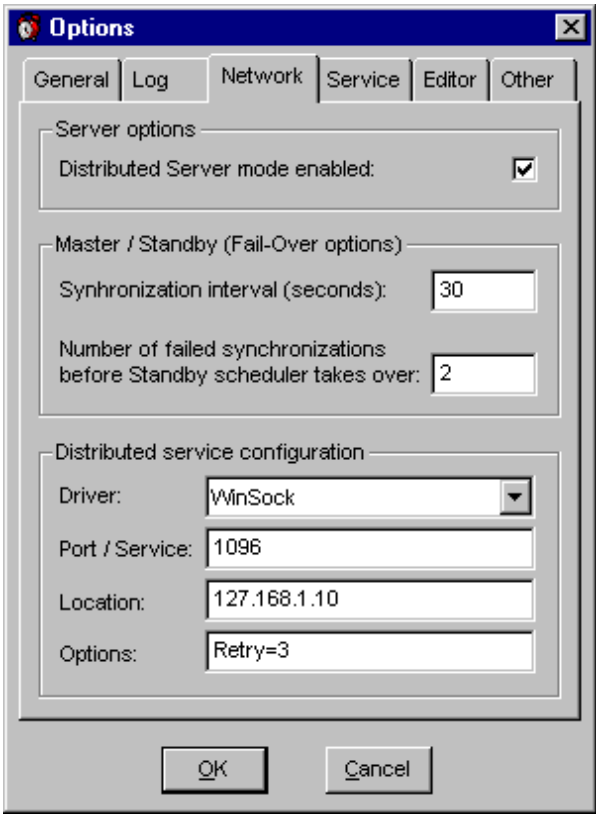
The "Fail-over Mode" is a “crash protection" mode in which the 24x7 Scheduler ensures high availability and error-recovery, reducing possible system downtime that can impede important processing functions, and even stop time-critical computing that is vital to your business.

The 24x7 Scheduler can run on two or more machines simultaneously, eliminating a single point-of-failure. However, only one scheduler at a time can serve as the Master Scheduler. All other schedulers run in Standby mode. When "Fail-over Mode" is activated, all information written to the Master Scheduler's job database is mirrored in the Standby Scheduler's database using sophisticated synchronization techniques. If the Master component becomes unavailable, the 24x7 Scheduler will perform an unattended rollover to the first Standby Scheduler to respond. This Standby Scheduler then becomes the new Master Scheduler. This architecture ensures that jobs will run on time in the event of a machine failure and that jobs will continue processing without interruption.

Optionally you can setup both Master and Standby Schedulers to run on the same machine. This will ensure continued processing should the Master Scheduler fail.

To enable "Fail-over Mode" :

1 Click **Tools** menu, then click **Options**. The 24x7 Scheduler options dialog box appears (see image below).
2 Select **Network** tab.
3 Check **Server mode** option.
4 For the Standby Scheduler specify **synchronization interval** parameter, which tells the 24x7 Scheduler at what interval you want the Standby Scheduler to connect to the Master Scheduler and request fresh job database snapshot. Note that a zero interval will disable multi-instance synchronization.
5 Enter connection parameters for distributed service.
6 Check **Trace enabled** option if desired. This option is used to troubleshoot Standby/Master Scheduler connections. When tracing is enabled, 24x7 Scheduler logs all activity between the Standby and the Master components to the STANDBY.LOG and MATER.LOG files as appropriate. This internal information can be helpful in debugging, including memory usage, an internal call trace, and the types and values of passed parameters are also stored. In addition, 24x7 Scheduler logs all activity to a console window. This helps simplify the troubleshooting process.
7 Restart 24x7 Scheduler.




**Tips:**

- Be careful about selecting the synchronization interval at which the Standby Scheduler connects to the Master Scheduler and requests a job database snapshot.

**See also:**

Setting Connection Parameters

Testing Connection to Master Scheduler

Starting Master and Standby Schedulers

# Connection Parameters for Fail-over Mode and for Remote Agents

**Driver** - The communications driver that will be used for the connection. Values are:

- WinSock
- NamedPipes
- Local

**Port/Service** - Meaning of this parameter is different for different communication drivers. For details refer to the following table:

| Driver | Parameter value |
|---|---|
| Winsock | Specify either of the following:<br>• The port number for the 24x7 Master Scheduler or Remote Agent (for example, 10099). 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.<br>• The service name for the 24x7 Master Scheduler. The service name is an indirect reference to the 24x7 Scheduler 's 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 WinNT. You may need to edit this file manually to add the 24x7 Scheduler Port or Service. |
| NamedPipes | 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. |
| Local | This property is ignored. |

**Location** - This specifies the location of the 24x7 Master Scheduler. The value of this parameter is different for different communication drivers. For details refer to the following table:

| **Driver** | **Parameter value** |
|---|---|
| Winsock | Specify either of the following:<br>• The IP address (for example, 199.99.99.91)<br>• The host name of the remote computer (network computer name in workgroup)<br>• LocalHost (This tells the 24x7 Standby Scheduler that the 24x7 Master Scheduler resides on the local machine.) |
| NamedPipes | 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. |
| Local | This property is ignored. |

**Options** - This specifies one or more additional communications options. If you want more than one option, you will need to separate the options with commas. This property is ignored for the **Local** driver.

| | |
|---|---|
| **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 Standby Scheduler will try to connect when the Master Scheduler'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 Standby and Master Schedulers must |

| | |
|---|---|
| | have the same setting. If there is a discrepancy between the Standby and Master Scheduler's settings, the communication will fail. Setting this option to 1 may improve performance slightly. Applies to the **WinSock** driver only. |

**See also:**

About Fail-over Mode

Testing Connection to Master Scheduler

Starting Master and Standby Schedulers

# Testing Connection to Master Scheduler

After setting up the connection parameters for both Master and Standby Schedulers, you will be able to test how well they communicate with each other:

1 Make sure both computers running Master and Standby Schedulers have been connected to the network.
2 Enable **Tracing** in the Program options on both systems.
3 Start 24x7 Master Scheduler. Activate the console window created by the Master Scheduler.
4 Start 24x7 Standby Scheduler. Activate the console window created by the Standby Scheduler.
5 Read messages on the Standby console. If you do not see the message "Connection successful", change the connection parameters on the Standby Scheduler and try again. If necessary, change parameters on the Master Scheduler.

If the intercommunication process succeeded, most GUI elements of the Standby Scheduler window become disabled and grayed out. The program title should have changed to reflect the "Standby" mode.

1 Shutdown the Master Scheduler computer.
2 Keep an eye on the Standby Scheduler. After the specified **synchronization interval** elapses, the Standby Scheduler will try connecting to the Master Scheduler. As a result, the connection will fail and the Scheduler will switch to the Master mode.

If the process succeeds, all previously disabled GUI elements of the Standby Scheduler become enabled. The program title should change to reflect the switch to the "Master" mode.

You may also want to watch for the Master Scheduler's console window during the connection process to see if there are any "Request rejected" messages.

 **Tips:**

- The console window cannot have a vertical scrollbar due to Windows® limitations. The number of visible lines (visible buffer size) is based on the font settings for the console window. You can change the font using the console window's control menu, **Properties** item.
- To see the entire intercommunication log, select **Tools** menu, then click **Log Viewer**. The Log Viewer window will open. See messages on **Master** and/or **Standby** tabs.

**See also:**

About Fail-over Mode

Setting Connection Parameters

Starting Master and Standby Schedulers

# Starting Master and Standby Schedulers

When starting up in Fail-over Mode, the 24x7 Scheduler on start up attempts to connect to the Master Scheduler. If that connection fails, the 24x7 Scheduler will immediately switch to the Master Scheduler mode, otherwise it will then automatically obtain a snapshot of the Master Scheduler's job database then switch over to the Standby Scheduler mode.

**Connection troubleshooting:**
Make sure that you have checked the **Distributed Server mode enabled** option in the system Options for both Master and Standby Schedulers. Also make sure you have setup the **Distributed Service Configuration** properly for both Master and Standby Schedulers. Make sure that both configurations use the same communication **Driver, Port** number and that the Standby Scheduler in the **Location** field points to the Master Scheduler. Click **Tools/Options** menu, click **Network** tab page to verify the configuration.

 **Tip:**
To implement cyclical Master/Standby switching and ensure uninterrupted job processing, you can point the Master Scheduler to the Standby Scheduler. To do this, in the system Options of the Master Scheduler specify the **Location** of the Standby Scheduler. Doing this means that in the event of the Master Scheduler software or hardware failure, the Standby Scheduler will switch to the Master mode. If you restart the original Master Scheduler, it will find the new Master Scheduler and will switch to Standby mode.

**See also:**

About Fail-over Mode

Setting Connection Parameters

Testing Connection to Master Scheduler

# Restricting Access to Master Scheduler

24x7 Scheduler supports several methods for securing access to 24x7 Master Schedulers.
**For complete list of methods see the Security topic.**

For compatibility with previous versions, 24x7 Scheduler still supports restricted access by serial number.
This help topic describes how to use and setup security based on the serial numbers.

**To restrict the list of Standby Schedulers that can connect to and take over a Master Scheduler:**

1 In the 24x7 Master Scheduler installation directory create a text file called RESTRICT.LST.
2 Edit this file in Notepad or any other text editor. Enter a list of 24x7 Scheduler serial numbers that can be used to access this Master Scheduler. Each serial number must appear on a separate line.

**To allow any 24x7 Standby Scheduler to connect to a Master Scheduler:**

1 Delete RESTRICT.LST file from the 24x7 Master Scheduler installation directory.

 **Important Notes**:
- Each installed copy of the 24x7 Scheduler must have a unique serial number. Please see license agreement for details.
- The list of allowed requestors RESTRICT.LST is also used to restrict 24x7 Schedulers that can submit jobs to a Remote Agent. You will need to remember this when switching between Master Scheduler and Remote Agent modes.

**See also:**

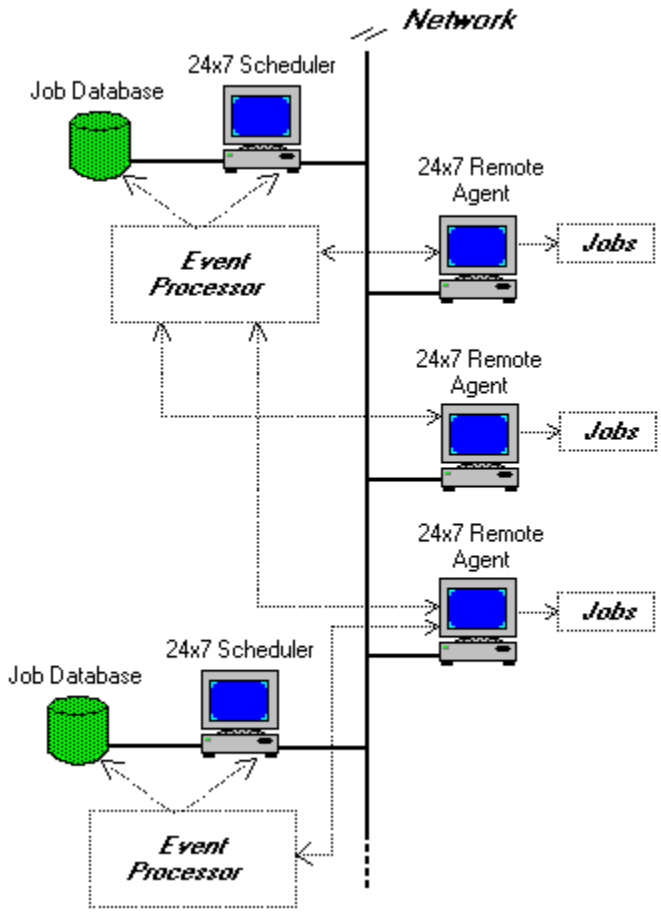
Security
About Fail-over Mode

# Chapter 7: Remote Agents

## About Remote Agents and Remote Jobs

The 24x7 Scheduler supports remote jobs. This means that you can run the main 24x7 Scheduler on one computer and execute jobs on a different computer connected to it via local or global network.

The 24x7 Scheduler Remote Agent provides a way to execute jobs on remote computers. The Remote Agent must be installed on each computer that you want processing to occur. Remote Agents are important parts of the 24x7 Scheduler distributed features. The 24x7 Scheduler distributed architecture allows a job created on the main scheduler computer to be executed by the Remote Agent on the remote computer. All job maintenance and event processing remain on the main 24x7 Scheduler computer. This allows you to have a single point of administration and scheduling for all distributed jobs.




**Important Notes:**

- Before starting a 24x7 Remote Agent for the first time you should run the 24x7 Scheduler on that remote machine in order to configure the 24x7 Scheduler network properties and communication parameters. See **Setting Connection Parameters** topic for details.
- Remote jobs can be also executed on computers running 24x7 Master Schedulers. Any 24x7 Scheduler or 24x7 Remote Agent can connect to other 24x7 Remote Agents and also to 24x7 Master Schedulers.

**What happens when it is the time to execute a remote job?**

The main 24x7 Scheduler (event processor) will attempt to connect to the 24x7 Remote Agent whose name is specified in the job properties. If the Remote Agent is not installed or it is not running on the selected computer, an

error will occur and the job will fail. The error code and text will be written to the job log. If the connection succeeded, the main 24x7 Scheduler will submit the job definition to the Remote Agent. If this job has the **asynchronous** option turned off, the main 24x7 Scheduler will wait while the job is being executed by the Remote Agent on the remote computer. If the job has the **asynchronous** option turned on, the main 24x7 Scheduler will post the job for asynchronous execution on the remote computer and immediately after that, continue normal job processing.

**See also:**

Starting Remote Agent
Remote Agent Profiles
Synchronous and Asynchronous Connections

# Starting Remote Agent

**To start the 24x7 Remote Agent from the command line:**

1 Change the current directory to the 24x7 Scheduler directory. For example: *CD "C:\Program Files\24x7"*
2 Run the command **24x7 /AGENT**. The 24x7 Remote Agent will then start.

**To create a shortcut on the Desktop:**
Alternatively, you may want to create a shortcut to start the 24x7 Remote Agent.

1 Right-click anywhere on the free area of the Desktop. A context menu will appear.
2 Click **New**, then click **Shortcut**.
3 Type the command line for the 24x7 Scheduler ending with the /AGENT parameter. The command line must include the full path to the **24x7.EXE**. For example *"C:\Program Files\24x7\24x7.EXE" /AGENT*.
4 Click the **Next** button.
5 Type the name for the newly created shortcut. For example: *24x7 Agent.*

Double-click on the shortcut icon. This will start the 24x7 Remote Agent.

 **Important Notes:** Before you start the 24x7 Remote Agent for the first time you should run the 24x7 Scheduler on the remote machine to configure 24x7 Scheduler network properties and communication parameters. See **Setting Connection Parameters** topic for details.

**To start the 24x7 Remote Agent each time Windows starts:**

1 Click the **Start** button, and then point to the **Settings**.
2 Click **Taskbar**, and then click the **Start Menu Programs** tab.
3 Click **Add**, and then click **Browse**.
4 Locate **24x7.EXE**, then double-click it.
5 Press **End** button on the keyboard, then type **/AGENT**.
6 Click **Next**, and then double-click the **StartUp** folder.
7 Type the name (such as “24x7 Agent" ) that you want to see on the **StartUp** menu, and then click **Finish**.

**See also:**

About Remote Agents
Example Remote Setup

# Restricting Access to Remote Agents

24x7 Scheduler supports several methods for securing access to 24x7 Master Schedulers and 24x7 Remote Agents. **For complete list of methods see the Security topic.**

For compatibility with previous versions, 24x7 Scheduler still supports restricted access by serial number.

This help topic describes how to use and setup security based on the serial numbers.

**To restrict access to a Remote Agent:**

1 In the 24x7 Remote Agent installation directory, create a text file called RESTRICT.LST.
2 Edit this file in Notepad or any other text editor. Enter a list of 24x7 Scheduler serial numbers that can be used to access this Remote Agent. Each serial number must appear on a separate line.

**To allow any 24x7 Scheduler to connect and submit jobs to a Remote Agent:**

1 Delete RESTRICT.LST file from the 24x7 Remote Agent installation directory.

**Important Notes**:

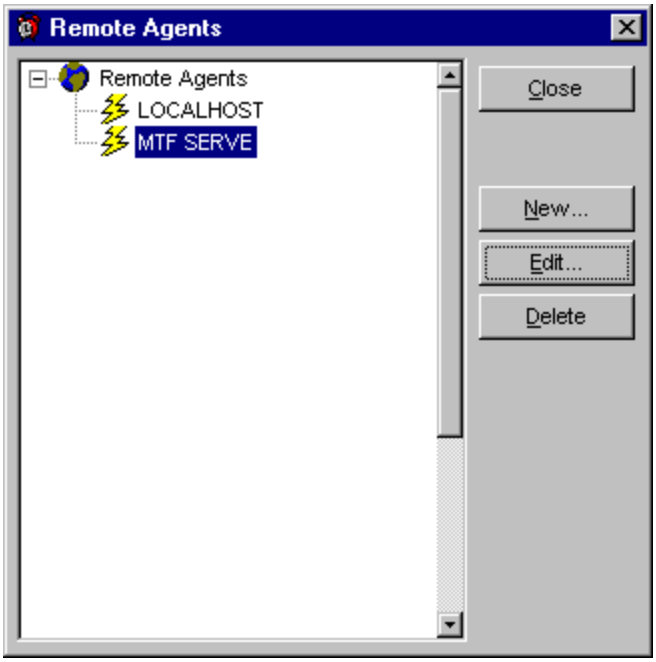
- Each installed copy of the 24x7 Scheduler must have a unique serial number. Please see license agreement for details.
- The list of allowed requestors RESTRICT.LST is also used to restrict 24x7 Standby Schedulers able to connect to the 24x7 Master Scheduler. Remember this when switching between Remote Agent and Master Scheduler modes.

**See also:**

Security
About Remote Agents

# Remote Agent Profiles

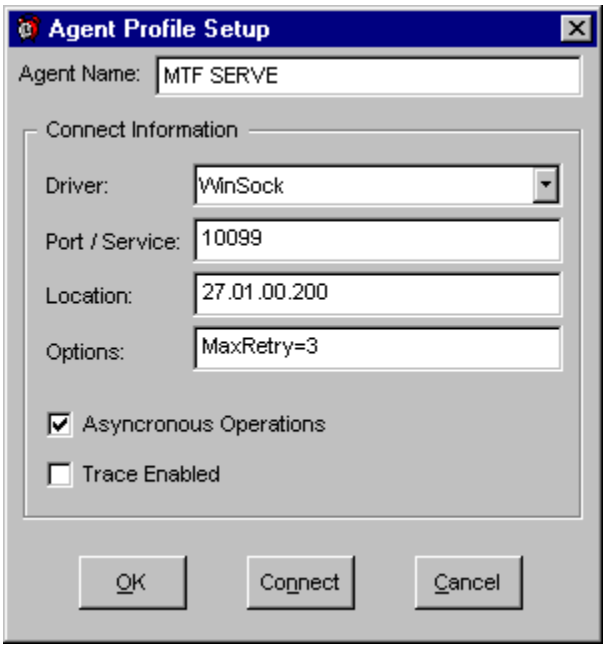
Before a job can be submitted to the **24x7 Remote Agent** or **24x7 Master Scheduler**, the Remote Agent profile must be created on the main 24x7 Scheduler. The main 24x7 Scheduler uses profile information when connecting to the 24x7 Remote Agent or 24x7 Scheduler. Each agent requires a separate profile.




To add, modify, and delete Remote Agent profiles, click the **Tools** menu, and then click **Remote Agents**. The remote agent profiles dialog box will appear.

**To add a new profile**:

1 Click the **New** button.
2 Enter Remote Agent connection information.
3 If the specified Remote Agent is running, click the **Connect** button to verify the connection parameters. If the connection fails, modify the connection information and try again.
4 Click the **OK** button.



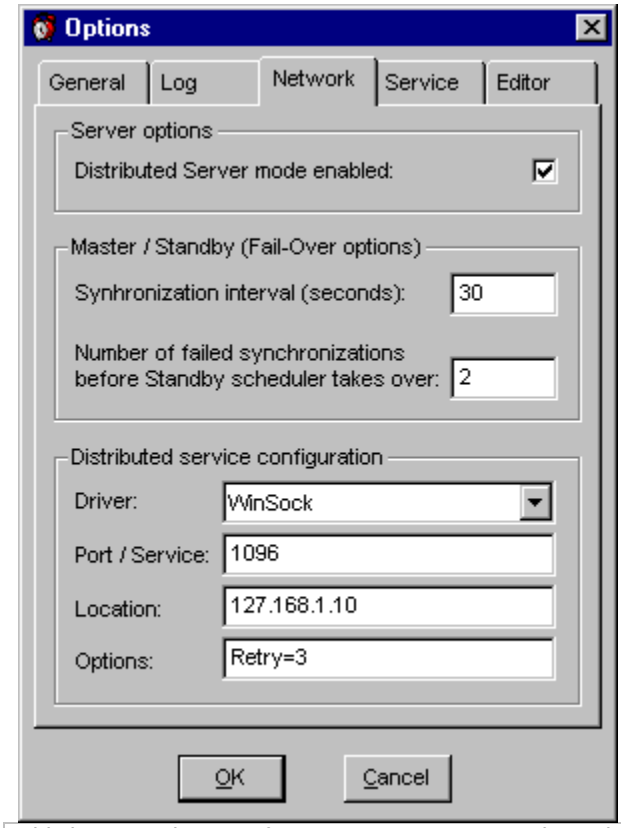

**To modify an existing profile**:

1 Select the desired profile in the **Remote Agents** list (see remote agent profiles dialog box above).
2 Click the **Edit** button.
3 Modify Remote Agent connection information.
4 If the specified Remote Agent is running, click the **Connect** button to verify the connection parameters. I if the connection fails, modify connection information and try again.
5 Click the **OK** button.

**To delete a profile**:
1 Select the desired profile in the **Remote Agents** list (see remote agent profiles dialog box above).
2 Click the **Delete** button.

**See also:**
About Remote Agents
Setting Connection Parameters
Example Remote Agent Setup

# Sample Remote Agent Setup




This is a sample setup for Remote Agent connection using TCP/IP protocol.

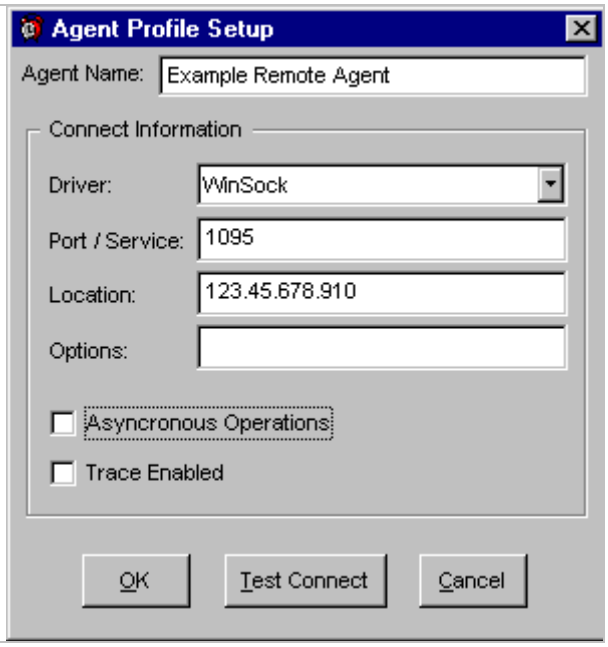
On the 24x7 Remote Agent computer:

1 Run the 24x7 Scheduler (normal mode).
2 Click **Tools/Options** menu, after the Options dialog box appears, select **Network** tab page.
3 Select **Winsock** driver. Enter **10095** for Port.
4 Click **OK**, choose **No** when prompted to restart the Scheduler then exit the Scheduler.
5 Start the 24x7 Remote Agent. The agent is now ready to accept requests.

You can now setup new Remote Agent profiles on the computer installed with the main 24x7 Scheduler

On the main 24x7 Scheduler computer:

1 Run the 24x7 Scheduler.
2 Select **Tools/Remote Agents** menu, after Remote Agents dialog box appears, click **New** button. The Agent Profile dialog box will appear.
3 Enter the desired Remote Agent name (any text up-to 30 characters long).
4 Select **Winsock** driver. Enter **10095** for port.
5 In the **Location** field specify IP address of the Remote Agent computer. Or, click **Test Connect** button to test connectivity to the Remote Agent.
6 Click **OK**, then click **Close** button on the Remote Agents dialog box.
7 Modify properties of the jobs that you want to execute on the remote computer. From the Host drop-down list select the name of the Remote Agent created earlier.




**See also:**
Remote Agent Profile
Connection Parameters

# Synchronous and Asynchronous Connections

The 24x7 Scheduler supports both synchronous and asynchronous communications between the main 24x7 Scheduler and 24x7 Remote Agents. For remote jobs, the 24x7 Scheduler chooses the communication mode that matches the execution mode specified in the job properties. If the job needs to be executed asynchronously, the main 24x7 Scheduler will communicate with the Remote Agent using asynchronous calls, otherwise it will call the Remote Agent synchronously. The Remote Agent will then also run the submitted job also synchronously.

Unlike synchronous calls (which force the main 24x7 Scheduler to wait until job processing has completed on the Remote Agent computer), asynchronous calls free the main 24x7 Scheduler to do other work while the Remote Agent processes the request.

When the Remote Agent receives a synchronous call, it executes the request immediately. The main 24x7 Scheduler will wait until processing has completed.

When the Remote Agent receives an asynchronous call, it adds the submitted job to a queue and performs the job processing at a later point in time. Meanwhile, the main 24x7 Scheduler will continue with other work while the Remote Agent handles requests.

All asynchronous requests are executed by the Remote Agent in the order they are received. However, the exact timing of the job execution cannot be guaranteed. If the Remote Agent receives a synchronous call after several asynchronous calls have been made, it will process the synchronous call as soon as possible. Therefore, queued asynchronous jobs will be processed after all synchronous jobs have been completed.

**See also:**
About Remote Agents
Setting Connection Parameters
Job Execution Properties

# Job Load Balancing

The 24x7 Scheduler supports "**Least busy**" load-balancing method for remote jobs. You can use **Host** and **Host List** job properties to specify that a job does not have a predetermined computer where it suppose to run. The **[Least busy]** value in the Host property indicates that the job is to be run on one of the computers specified in the Host List property. Before running such job 24x7 Schedule automatically determines which host is least busy and submits the job to that computer. 24x7 Scheduler takes into account all jobs that run on the specified computers including jobs that were submitted by other 24x7 Schedulers or submitted interactively by users utilizing 24x7 Remote Controls


For the Host List enter comma-separated name list of Remote Agents that can run the job. Order of agents in the list is not important. 24x7 Scheduler queries all agents before deciding which one is least busy.

**See also:**

About Remote Agents
Setting Connection Parameters
Job Execution Properties

# Chapter 8: Database Jobs

The 24x7 Scheduler allows you to create jobs that can access various database systems. The 24x7 Scheduler also provides easy access to corporate information stored in a wide variety of databases.

The 24x7 Scheduler can connect to a database through the ODBC interface or through a native database interface. The 24x7 Scheduler software includes a number of native database interfaces as well as numerous ODBC drivers. If you cannot find an appropriate ODBC driver, you can install and use one of the drivers included to standard 24x7 installation package.

Before connecting the 24x7 Scheduler to your database, you will need to do some preparatory steps. The following are the basic steps you should follow when preparing the 24x7 Scheduler to work with your database:

1 Get ready to use your database (start database server, start database listener, etc…)
2 (Optional) Install the ODBC driver or native database driver.
3 (Optional) Define the ODBC data source.
4 Create the database profile.
5 (Optional) Troubleshoot the database connection.

## Preparing to use your database

Preparing the database ensures that you will be able to access and use your data. The requirements differ for each database but, in general, preparing a database involves the following steps:

1 If network software is required, make sure it is properly installed and configured at your site and on the client machine.
2 Make sure the required database server software is properly installed and configured.
3 Make sure the required database client software is properly installed and configured on the client workstation. (Typically, the client workstation is the one running the main 24x7 Scheduler or the 24x7 Remote Agent.)

**Important:** You must install the appropriate client software for your database server version and operating system platform. See your database vendor for information.

## Installing the ODBC driver or native database driver

To connect the 24x7 Scheduler to your database, you must install the ODBC driver or native database interface that accesses the database. Select the desired driver or database interface when prompted to do so by the Setup program.

**Important**: If you are installing an ODBC driver, make sure you also install the ODBC interface on your computer. The ODBC interface is installed by default with 24x7 Scheduler.

## Defining the ODBC data source

Data that you access through an ODBC driver is referred to as an ODBC data source. An ODBC data source consists of the data and associated DBMS or file manager, operating system, and (if present) network software. When you define an ODBC data source, you provide information about the data source that the driver needs to connect to it. (Defining an ODBC data source is also referred to as configuring the data source.) You can use the ODBC Manager software to create and modify ODBC data sources. To start the ODBC Manager you will need to do either of the following:

**From the 24x7 Scheduler**

1 Click **Tools** menu, then click **Database Profiles**. The Database Profiles dialog box will appear.
2 Click the **ODBC** button.

**From Windows Control Panel**

1 Click the Windows **Start** button.
2 Select **Settings** menu, then select **Control Panel**. The Control Panel window will appear.
3 Double-click the **ODBC** icon.

**Completing the ODBC setup dialog box**

Define an ODBC data source by completing the ODBC setup dialog box for the ODBC driver you have previously selected to access the data source. The content and layout of the ODBC setup dialog box will vary for each driver, but most ODBC setup dialog boxes require you to supply the following information:

- Data source name and location,
- Data source description (optional),
- Other DBMS-specific connection parameters.

After you have created a data source, you can use it in the database profile already created in the 24x7 Scheduler.

# Creating database profiles

To create a new database profile:
1 Click **Tools** menu, then click **Database Profiles**.
2 Follow instructions described in the Database Profiles topic.

# Troubleshooting the database connection

The 24x7 Scheduler provides two tools for tracing database connections in order to troubleshoot problems:

- **Database Trace** - The Database Trace tool records the internal commands that the 24x7 Scheduler performs while communicating with a database. Database Trace writes its output to a file named PBTRACE.LOG located in the Windows home directory. You can view the contents of the log at any time by using the Log Viewer. To enable database tracing, select **Database Trace** options on the 24x7 Scheduler's options dialog (Select **Tools** menu, then click **Options.** An Options dialog box will appear.)
- **ODBC Driver Manager Trace** - The ODBC Driver Manager Trace tool records information about the ODBC API calls made by the 24x7 Scheduler while connected to an ODBC data source. The ODBC Driver Manager Trace writes its output to a file named SQL.LOG (by default) located in the Windows home directory or to a log file that you specify. You can view the ODBC Driver Manager Trace log at any time by using any text editor.

**See also:**
Database Interfaces
ODBC Interface
Database Profiles
Testing Connection to Your Database

# Chapter 9: Job Automation Scripts

The 24x7 Scheduler provides you with two robust scripting languages: Visual Basic Script (VBScript or VBS) and Job Automation Language (JAL). Both languages give you the ability to customize the behavior of any scheduled job. Using scripting you can create very complex conditions for the job triggers, implement powerful error-checking and error-handling, create customized Notification Events and Actions. For example, if you want to setup a job that runs every 15 minutes on workdays only, you can create a script job and schedule this job to run **all day** every 15 minutes. In the job script, you would code the condition that checks whether the current time is between 9:00 AM and 5:00 PM and the day is a workday. If these conditions have been satisfied, you will launch the desired process; otherwise you will exit the script.

Another typical example of using script is to search common error messages in a log file created by the previously executed program.

The 24x7 Scheduler also provides you with a powerful integrated editor that features JAL, VBS and SQL syntax highlighting, bookmarks, search and replace, context help for JAL statements and much more...

**See also:**

- Help for Visual Basic Script
- Job Automation Language Overview
- Job Automation Language Examples

# Chapter 10: Job Activity and System Events Logs

Job Activity and System Events Logs are optional, however, it is highly recommended that you enable logging options for mission-critical jobs. The job log provides a complete audit trail for all job activities. You can use the information stored in the log to troubleshoot incorrect or “not on-time” job execution, and resolve scheduling conflicts between different jobs.

The 24x7 Scheduler writes to the log the date and time at which an event occurred, the event severity, job ID, job name, and event description. For an error event, the description includes job status, error code and complete error message. All errors fall into four categories:

- **Operating System errors –** These errors are most likely to occur while running external programs and documents. They are reported by the Operation System. Refer to Windows documentation for a complete description of Operation System errors.
- **Database errors –** These errors occur while performing database operations. They are reported by the DBMS. Refer to your database documentation for a complete description of database errors.
- **Program errors –** These errors occur while running external programs and documents. They are reported by the scheduled programs. Refer to the program documentation for a description of program errors.
- **Scheduling errors –** These errors occur because of incomplete or invalid job definitions. They are detected and reported by the 24x7 Scheduler job execution engine. Check and correct invalid job definition if these errors occur.

Contact SoftTree Technologies Technical Support for assistance If you are unable to resolve the error you are experiencing.

## Supported Log Files

| Log File | Description |
|---|---|
| schedule.log | This is the main log file containing all system level events and job activity audit trail. |
| HTML log files | This is a group of files that all together contain the same records as the main event log schedule.log. These files are updated only when the HTML Status Report option is enabled. |
| Windows NT event logs | Windows NT event logs may contain a copy of all records from the main event log schedule.log as well as all events and errors reported by various 24x7 extensions, including Macro Recorder, SNMT Extension Agent and many other. Job audit records are written to the NT Application Event Log if the "Logging to NT event log" option is enabled. This option has no effect on other sources of event records. |
| jdl.log | This log file contains records written by various utilities that either explicitly or implicitly use JDL interface to create, modify and describe active jobs. This log is maintained if the "Tracing" option is enabled |
| pbtrace.log | This log file contains tracing of low level database operation generated by 24x7 while executing database jobs and notification events. This log is maintained if the "Database Tracing" option is enabled. |
| master.log | This log file contains tracing of low level calls generated by 24x7 Master Scheduler (calls made to remote 24x7 components and events recorder by the Master scheduler network listener). This log is maintained if the "Tracing option" is enabled. |
| agent.log | This log is similar to master.log. It is updated by |

| | |
|---|---|
| | 24x7 Remote Agents. This log is maintained if the "Tracing option" is enabled. |
| rcontrol.log | This log is similar to master.log. It is updated by 24x7 Remote Control utility and various 24x7 Remote Control APIs. This log is maintained if the "Tracing option" is enabled. |
| intrface.log | This log contains audit trail for 24x7 DDE interface. It is updated when the DDE API is used both directly through DDE API and indirectly through various 24x7 utilities that internally use the DDE API. This log is maintained if the "Tracing option" is enabled. |
| ftp.log | This log contains audit trail for all FTP and Secure FTP (SFTP) operations. This log is maintained if the "Tracing option" is enabled. |
| telnet.log | This log contains audit trail for all Telnet and Secure Shell (SSH) operations. This log is maintained if the "Tracing option" is enabled. |
| sync.log | This log contains complete audit trail for all operations involving file replication methods ( SyncDir, SyncRemoteDir, SyncFTPDir). |
| script.log | This log contains tracing of the last script type job. This log is maintained if the "Tracing option" is enabled. For complete job tracing see <nnn>.log files in the Performance Data directory. |
| <nnn>.log files in Performance Data directory | <nnn> logs contain complete tracing for all script type jobs. 24x7 maintains a separate log file for each job. <nnn> is a placeholder for the job ID. <nnn> logs are maintained if the "Tracing option" is enabled. |

**See also:**

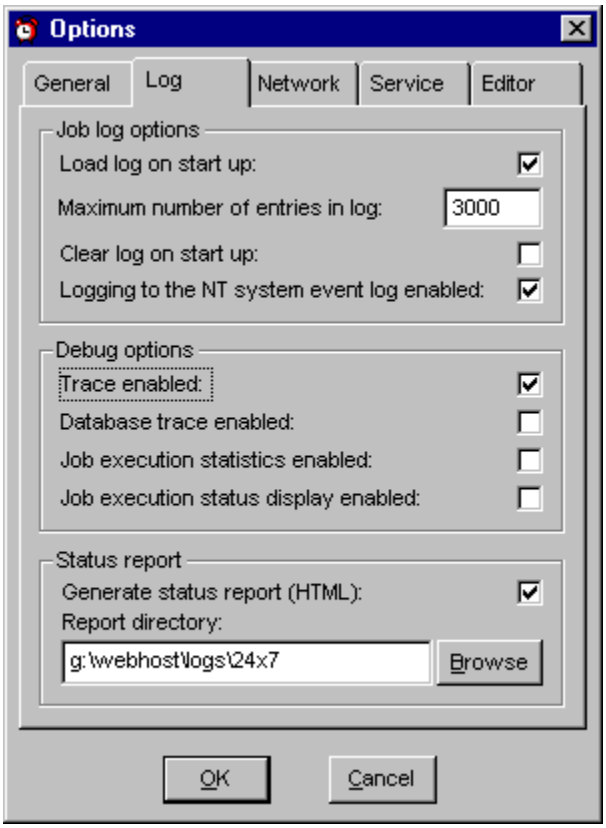
- Execution Log
- Log Viewer
- Trace Features
- Job Execution Statistics

# Chapter 11: Reports

24x7 Scheduler supports a number of different job performance reports and the self-updateable real-time job **Status Report**

## Status Report

The 24x7 Scheduler allows you to generate real-time HTML reports that you can use to check historical job activity and monitor the status of currently running jobs. Moreover, you can configure the 24x7 Scheduler to automatically update and upload these reports to a Web server (via Intranet or Internet). A Web browser can then be used to view these reports from virtually anywhere.




To enable generation of the Status Reports:

1 Select **Tools** then **Options** menu.
2 Check the **Generate status report** option.
3 Type the destination directory in the **Status report directory** field. Alternatively, you can click the **Browse** button to select the desired directory. If you leave the field blank, the 24x7 Scheduler will save reports in the installation directory.
4 Click the **OK** button

**Tips:**

- For the **Status report directory,** you can choose the directory on your Web server. In this case, all status report changes will be available immediately (real-time) to Internet users. Alternatively, you may want to setup a job that will upload created HTML files to the desired Web server. That job can be run every 30 minutes or at another specified interval providing near real-time reporting over the Internet.
- 24x7 Scheduler optionally can generate consolidated status reports that are based on logs obtained from multiple 24x7 Schedulers and 24x7 Remote Agents. Use **Build consolidated job status reports** job template to create a job that will periodically download remote logs and generate consolidated status reports. In order to be to download remote logs the remote 24x7 Schedulers must be running in the distributed server mode (in other words in the **Master** mode).

- You should setup some security restrictions on your Web server if you do not want the report to be available to all Internet users.

**See also:**

Using Web Browser to View Status Report
Job Monitor

# Using Web Browser to View Status Report

To preview the Job Status Report in the default Web browser., click **View** menu then select **Status Report**. If you have configured the 24x7 Scheduler to update Status Report on your Web server, you can also browse it on the Web server.

 **Tips:**

- You can use your Web browser "find" feature to quickly find the desired job.
- You can also use your Web browser to print the Status Report.

# Job Performance Reports

# Reports Overview

There are a number of statistical reports available in 24x7 Scheduler. Each report can be viewed and printed from the Report window. These reports include information from the job database and from files containing previously collected job performance data. The following reports are available:

- Job Forecast
- Job Resource Consumption
- Job Performance & Trends
- Job Timing and Conflicts
- Job Queue Utilization
- Server Load Balancing
- Job Database Summary

Use can use them to forecast jobs, spot job trends, tune and troubleshoot jobs, and provide others with information through printing.

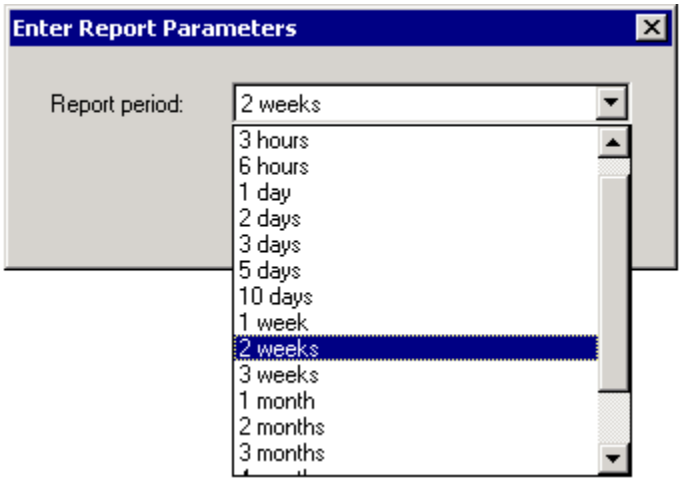
 **Tips:**

- Most reports use job performance data for job processing analysis. This data is not available automatically. To allow job performance data collection you must enable **Job Execution Statistics** option in the 24x7 system options. Do not run repots immediately after you enable this option. Wait as long as necessary for each job to run at least 4-5 times. This will allow 24x7 Scheduler to gather sufficient volume of job performance data for an accurate job analysis. Do not force/trigger jobs manually for the purpose of performance data collection, as it will lead to inaccurate reports. The accuracy of job reports is proportional to the volume of gathered data. Longer you wait , more jobs run, and more valid job performance data can be gathered.
- Job performance data is stored in JPD files located in the **Performance Data** subdirectory. 24x7 Scheduler creates separate performance file for every job. JPD files are regular ASCII text files. Size of each JPD file is proportional to the number of job runs and so it grows faster for jobs that run frequently. If the reports become very, very slow and take a long time for the analysis and charting, it may indicate that you have large volume of job performance data collected. Check file sizes in the **Performance Data** subdirectory. If There exist files over

several hundreds Kbytes large you should either delete them completely or better purge old data from these files to make them smaller. You can use Windows Notepad or other text processor for this purpose.

### To view a report:

1 In the Report menu, choose the type of report you want. The report window will appear.
2 If the selected report analyses job performance data, you will be prompted to choose time interval for the analysis. Select the desired interval from the drop-down list and click OK.




3 You can use the Zoom feature to zoom in and out. You can either select one of the predefined values in the zoom drop-down box or type in the value you want then press the Enter key.

### To print a report:

1 Run report as described above.
2 When the report is ready, click the Print button or select File/Print command from the Report window menu

### To re-run a report:

1 Click the Re-run button or select File/Re-run command from the Report window menu.
2 If the selected report analyses job performance data, you will be prompted to choose time interval for the analysis. Select the desired interval from the drop-down list and click OK.

## Job Forecast Report

This report calculates 7-day job forecast for all types of job, including jobs with event-based schedules. Forecast for time-based jobs is based on job schedule settings. Forecast for other, non-time-based jobs is based on statistical analysis of job performance data (if that data is available). In other words, forecast for jobs having non-time-based schedule is extrapolated from collected job performance data statistics. Such forecast is not guaranteed to be accurate and it is provided solely for the purpose of helping you to identify which jobs you can expect to run during the forecast period.

Frequently running jobs with less than 15 minutes recursion interval are not shown on this report in order to improve the report's readability. You would know about these jobs anyway.

The report presents data by day of week, sorted by job forecasted start time

The following columns are displayed on the report

**Job ID** – The job ID in the job database

**Job Name** – The job name in the job database

**Folder Name** – The name of the folder in the job database where the report is stored

**Start Time** – Job forecasted start time

**Schedule Type** – The job schedule type such as "daily job", "monthly job", "file-watch", "process-watch", and so on

**Notes:**
The Job Forecast Report is also available in 24x7 Remote Control. Since the **24x7 Remote Control** does not have access to the performance data for the remote 24x7 Master Scheduler, the report is calculated only for time-based jobs and this can make it differ from the same report if it is run on the 24x7 Master Scheduler computer.

**See also:**
Reports Overview
Job Monitor

# Job Resource Consumption Report

This report is based on statistical analysis of job performance data. **This report is available on Windows NT/2000/XP/2003/VISTA/2008/7 platforms only**.

The report consists of two parts: the summary chart for total system load by all jobs over selected report period and detail part for system load by individual job over selected report period. You can use this report for the following purposes:

1 To identify jobs consuming lots of system resources.

2 To better distribute and balance load on your system so the resource intensive jobs do not run at the same time. This can allow you to improve overall system performance and eventually allow more jobs to be run on the system.

The chart is scaled to fit the selected report period. You can try different report periods to see how the system is loaded over different periods of time. **Be aware that selecting large report period allows more job runs to be analyzed and thus it improves the report accuracy, but it can also increase calculation time.**

The following columns and values are displayed on the report

**CPU Usage (chart)** – Total CPU time spent for executing jobs

**Time (chart)** – System time

**Job ID** – The job ID in the job database

**Job Name** – The job name in the job database

**Folder** – The name of the folder in the job database where the report is stored

**Total Executions** – Number of times the job was executed during the selected report period

**Average CPU Time** – The average CPU time spent per job run during the selected report period

**Total CPU Time** - Total CPU time spent per job during the selected report period

**Tips:**
It is a good idea to run **Job Timing and Conflicts** and **Server Load Balancing** reports that together with Job Resource Consumption Report will provide you with a complete picture of the system load and thus allow you to figure out how to use your system more efficiently.

**Note:**
Failed jobs and jobs whose execution took less than 1 second of CPU time are NOT included on this report

**See also:**
Reports Overview
Job Timing and Conflicts
Server Load Balancing

# Job Performance and Trends Reports

This report calculates charts for top 20 jobs whose run-time duration have changed during the selected report period. This report is based on statistical analysis of job performance data.

You can use this report for the following purposes:

1 To identify jobs whose run-time duration significantly changed from run-to-run during the chosen report period.

2 To determine job duration trends (i.e., how quickly the duration changes over time) and thus you can use this information forecast future durations.

You should choose meaningful time interval for the report to allow the desired jobs to run for at least 4-5 times. Keep in mind that trend chart accuracy for each job highly depends on number of successful job runs during the chosen report period.

Jobs with less than 3 executions during the chosen report period are ignored when the report is built.

The chart is scaled to fit the selected report period. You can try different report periods. **Be aware that selecting large report period allows more job runs to be analyzed and thus it improves the report accuracy, but it can also increase calculation time.**

The following values are displayed on the report

**Duration** – Job duration in seconds

**Time** – System time

**Legend –** Chart legend that shows up to 20 jobs presented on the chart. Each legend label contains job ID job name.

Colored solid lines on the chart indicate actual job durations over time. Colored dashed lines on the chart indicate job duration trends. Job trend lines are built using simple linear regression model. Both actual and trend lines are drawn using the same unique color assigned to each job.

**See also:**
Reports Overview

# Job Timing and Conflicts Report

This report calculates Gantt chart for all jobs that ran during chosen report period. This report is based on statistical analysis of job performance data.

You can use this report for the following purposes:

1 To identify concurrent or conflicting jobs, those that happen to run at the same time or delay other jobs. This information can help you in job troubleshooting.

2 To identify jobs usually waiting long time in their job queue before being executed because their queue is busy executing other jobs.

3 To get the overall picture of how long it takes for jobs to be executed after they are queued.

4 To determine time windows when the system is not busy executing jobs or executing too many jobs simultaneously. You can use this information for the system load balancing.

5 This report also complements **Job Resource Consumption** report and both reports can be used to identify jobs that cause high system load. However, keep in mind that the Job Timing and Conflicts report charts jobs queue and executing times, and NOT job CPU usage time. Long running jobs do not always take a lot of system resources. For example, a database job can send some query to the database server and spend a long time "slipping" and waiting for the database response. In this case, most if not all processing happens actually on the database server computer, while the job consumes very little system resources on the scheduling computer.

The chart is scaled to fit the selected report period. You can try different report periods. **Be aware that large report periods may prevent you from seeing jobs whose duration and/or time in queue are very small as compared to the chosen report period. If the chart appears empty try selecting smaller report period.**

The following values are displayed on the report

**Time** – System time

**Job ID** – The job ID in the job database

**Legend** – The chart legend describing that job queue times are displayed using bright green color, job execution times are displayed using dark green color.

**See also:**

Reports Overview
Job Resource Consumption

# Job Queue Utilization Report

This report is based on statistical analysis of job performance data. The report contains two types of queue utilization charts: a chart for average time that jobs wait in queue before they are executed and a chart for number of queued jobs. Both charts differently show utilization of each active job queue over a period of time.

The main purpose of this report is to help you identify overloaded queues causing jobs to wait before they are executed.

Each chart is scaled to fit the selected report period. You can try different report periods. **Be aware that selecting large report period allows more job runs to be analyzed and thus it improves the report accuracy, but it can also increase calculation time.**

The following values are displayed on the report

**Time** – System time

**Average Time in Queue** – The average number of minutes that the queued jobs have to wait before being executed.

**# of Queued Jobs** – The number of simultaneous job in the queue. This number includes both pending and running jobs.

**Legend** – The chart legend shows active job queues and their assigned colors. Active queues are these that were used during the chosen report period.

**See also:**

Reports Overview
Job Resource Consumption
Job Timing and Conflicts
Server Load Balancing

# Server Load Balancing Report

This report is based on statistical analysis of job performance data. This report can be helpful if you have numerous jobs running by **24x7 Remote Agents**. The Server Load Balancing Report can help you identify 24x7 Remote Agents that carry more load than other. You should run this report on the main scheduling system, which is used to submit jobs to these Remote Agents.

The report contains two types of load charts: a chart for the load by day of week and a chart for the load by time of day. Both charts show utilization of each active Remote Agent as percent of all jobs executed during the chosen report period.

The main purpose of this report is to help you identify overloaded and under-loaded Remote Agents. You can use this information to better balance your system load.

Each chart is calculated using job performance data for the chosen report period. You can try different report periods. **Be aware that selecting large report period allows more job runs to be analyzed and thus it improves the report accuracy, but it can also increase calculation time.**

The following values are displayed on the report

**Time** – System time

**% of All Jobs** – The relative number of concurrent jobs running in a given point of time on a given computer. This is calculated as a percent value (the formula is: 1 – number of concurrent jobs over total number of jobs that were run during the chosen report period).

**Legend** – The chart legend shows active 24x7 Remote Agents and their assigned colors. Active 24x7 Remote Agents are these that were used during the chosen report period.

 **Important note:**
In the current version, the Server Load Balancing Report does not take into account jobs that were submitted to 24x7 Remote Agents from other computers. If this situation applies to your environment, the produced report can be inaccurate.

**See also:**

Reports Overview
Job Resource Consumption
Job Timing and Conflicts

# Job Database Summary Report

This report provides summary information about your job database. It consists of the report header and the report body. The header contains various totals by job type, job state and schedule type, and the body contains job list by job owner.

The following columns are displayed on the report

**User Name** – Job owner. This is the same as the Modify User property displayed on the Job Property Page in the Job Explorer

**Job ID** – The job ID in the job database

**Job Name** – The job name in the job database

**Job Folder** – The name of the folder in the job database where the report is stored

**Protected** – This indicates whether the job is protected

**Disabled** – This indicates whether the job is disabled

**Schedule** - The job schedule type such as "daily job", "monthly job", "file-watch", "process-watch", and so on

**Modified** – The date and time when the job was modified last time

**File Name** – The job database file name. If the report is run in the 24x7 Remote Control, this value may be invalid.

**File Time** – This is the date and time when the job database file was updated last time. If the report is run in the 24x7 Remote Control, this value may be invalid.

**See also:**

Reports Overview

# Chapter 12: Exception Dates

The 24x7 Scheduler allows you to specify some “exception” dates (e.g. holidays) when you do not want a job to run. This option only affects jobs set to run daily, weekly, or monthly. If the job is triggered by another event, such as the arrival of certain files or e-mail message, it will run even if the event was triggered on a holiday.

When scheduling a job that must not run should it happen on a holiday, make sure to select the **Skip Holidays** option in the job’s properties. For example, if the program is scheduled to run every Friday and this option is checked, the program will not run on holiday Fridays such as Independence Day.

Alternatively, you may want to select the **Slide Holidays** option (monthly jobs) which instructs the 24x7 Scheduler to avoid running this particular job when it falls on a holiday and postpone the job’s execution until the next workday. For example, if the program is scheduled to run every Friday and this option is checked, the program will not run on a Friday if it falls on Independence Day and will run it the following Monday instead.

The holiday list can be altered to suit your needs. It may include any dates that you want to exclude from normal processing. The holiday list is shared by all jobs. If you want to have a different list for a job or group of jobs, you can install a second copy of the 24x7 Scheduler to a different directory. The holiday list is stored in the HOLIDAY.TXT file located in the 24x7 Scheduler installation directory. Another way of having different exception days for different jobs is to use jobs written in 24x7 Script. Check out examples of this kind of schedule provided with the standard 24x7 installation package.

If you want to modify the list of holidays, you will need to use the Holiday Editor. To launch the Holiday Editor, select **Tools** menu, and then click **Holidays**. Not all holidays have fixed dates. You should update the holiday list every year or fill out this list for a few years ahead.

**See also:**

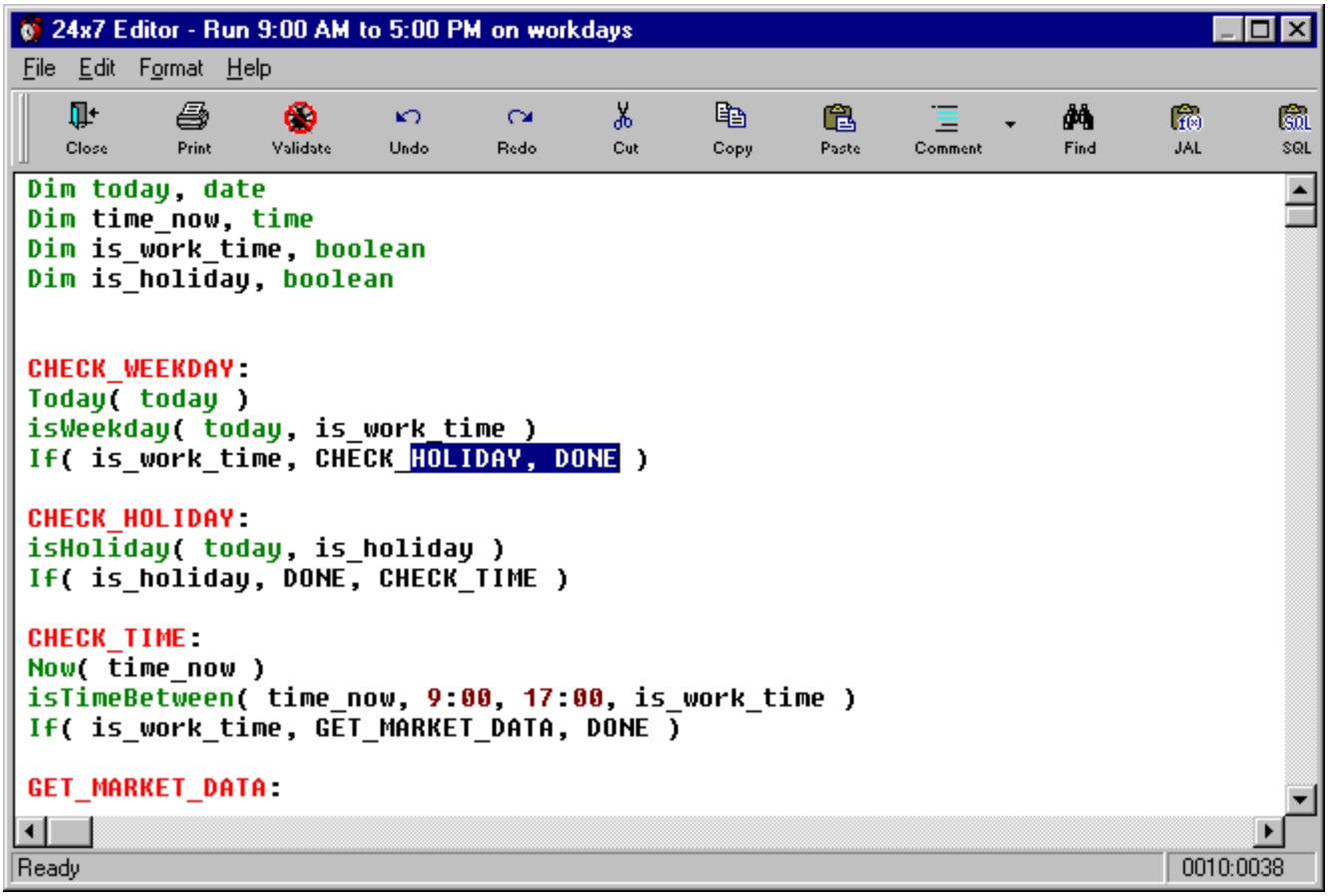
Installation and Uninstallation
Job Schedule

# Chapter 13: Editor

**Description**

The 24x7 Scheduler incorporates a powerful editor that enables you to edit your JAL, VBS and SQL scripts efficiently. It includes important editing features such as: syntax highlighting, Script Assistants - Paste JAL and Paste SQL syntax, search and replace, virtually unlimited number of Undo and Redo levels, setting bookmarks, as well as other standard editing functions.

You can use the Editor to write JAL and VB scripts for 24x7 Script jobs. The Editor can also be used to write SQL commands for Database-type jobs. The editor will automatically provide appropriate syntax highlighting.




**Using the Editor**

To create or modify a script for a job, open the Job Wizard for that job then click the **Next** button. Click the **Editor** button.

**Entering JAL, VBS and SQL scripts**

You can enter a script in three ways:

- pasting the text from the clipboard,
- typing the text in the Editor's workspace,
- importing a file containing the script.

**Searching in Scripts**

1 Click the 

**Find** button on the Editor window toolbar or select **Edit/Find** command from the menu. The **Find** dialog box will open.
2 Type your search string in the **Find What** edit box.
3 Specify the direction of the search operation by selecting either the **Up** or **Down** radio-button.
4 If you want to perform a case-sensitive search, then activate the **Match Case** check box.
5 If you want to search for whole words only, then activate the **Match Whole Word Only** check box.
6 Click the **Find Next** button to start the search. The 24x7 Scheduler will highlight the first instance of the search string it finds. To continue searching, click the **Find Next** button again.

**Replacing in Scripts**

1 Select the **Edit/Replace** command from the Editor's menu. The Replace dialog box will open.
2 Type your search string in the **Find What** edit box.
3 Type the replacement text in the **Replace With** edit box.
4 If you want to perform a case-sensitive search, then activate the **Match Case** check box.
5 If you want to search for whole words only, then activate the **Match Whole Word Only** check box.
6 To start the search and replace operation:
- If you want to scroll through the script and examine each instance of the search string before placing it, click the **Find Next** button. The 24x7 Scheduler will highlight each search string that it finds. To replace the search string, you must click the **Replace** button. To continue searching and replacing, you must repeat the steps above for each search hit.
- If you want to replace all instances of the search string without pausing, then click the **Replace All** button.

**Go to Command**

To go to a line:
1 Select the **Edit/Go to Line** command from the Editor's menu. The Go to Line dialog box will open.
2 Type the line number in the edit box.
3 Click the **OK** button to jump to the specified line.

To go to a previously set bookmark:
1 Select the **Edit/Go to Bookmark** command from the Editor's menu. The 24x7 Scheduler moves the edit caret to the line where the bookmark was set

**Importing and Exporting Scripts**

The 24x7 Scheduler stores all scripts in the job database. You can use Import and Export features to save and read scripts as ASCII files.

To export a script:
1 Select the **File/Export** command from the Editor's menu. The Save As dialog box will appear.
2 Specify the name of the file in which you want to save the script.
3 Click the **OK** button.

To import a script:
1 Select the **File/Import** command the Editor's menu. The File Open dialog box will appear.
2 Specify the name of the file from which you want to load the script.
3 Click the **OK** button.

*Warning: The contents of imported file will completely replace the current script in the Editor.*

**Printing Scripts**

To print a script:

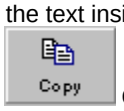
1 Click the 

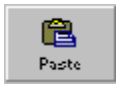
**Print** button on the Editor window toolbar. The Print dialog box will open.

2 Select the **Printer** from the drop-down list. It should contain a list of local and network printers that you can access. If you do not see any listed, then your computer is not configured for any printers.
3 Specify the desired print properties.
4 Click the **OK** button to print the script.

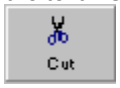
### Copying, Pasting and Cutting Text

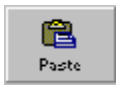
The Editor supports standard edit function such as Cut, Copy and Paste commands that move selected text to and from the Windows clipboard.

To copy and paste text:
1 Highlight the text inside the Editor window.
2 Click the 

**Copy** button on the Editor's toolbar. This action causes the selected text to be copied to the clipboard.
3 Place your cursor at the place where you want to paste the text. Click the 

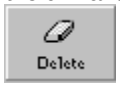
**Paste** button. The text will be copied onto the script.

To cut and paste text:
1 Highlight the text inside the Editor window.
2 Click the 

**Cut** button on the Editor's toolbar. This action causes the selected text to be copied to the clipboard and removed from the script.
3 Place your cursor at the place where you want to paste the text. Click the 

**Paste** button. The text will be copied onto the script.

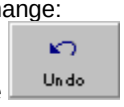
### Deleting the text

To delete the text:
1 Highlight the unwanted text inside the Editor window.
2 Click the 

**Delete** button on the Editor's toolbar. The 24x7 Scheduler will remove the highlighted text from the script.

### Undo/Redo Changes

The Editor supports up to 256 levels of undo/redo actions. The Undo action cancels the last edit, restoring the text to the content before the last change.

To undo a change:
1 Click the 

**Undo** button on the Editor's toolbar.

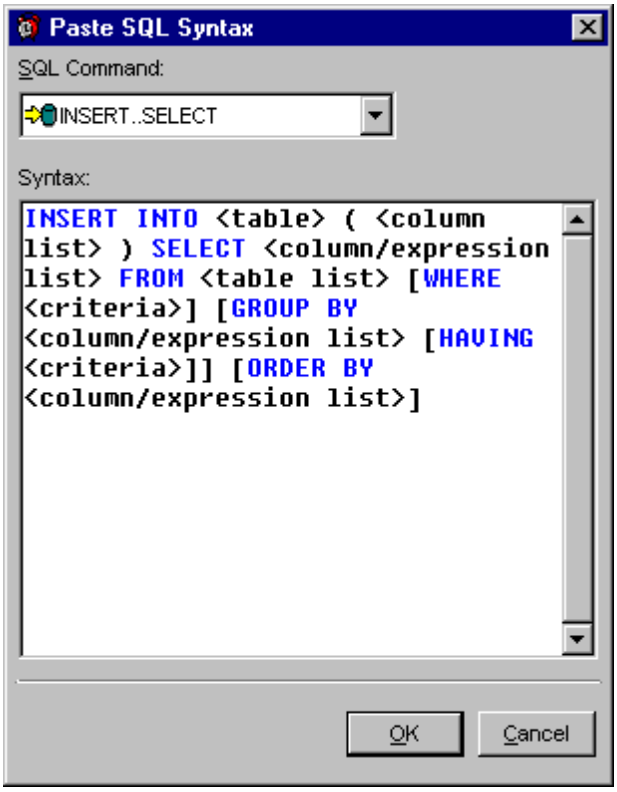
To repeat the last undone change:
1 Click the 

**Redo** button.

### Pasting SQL Syntax

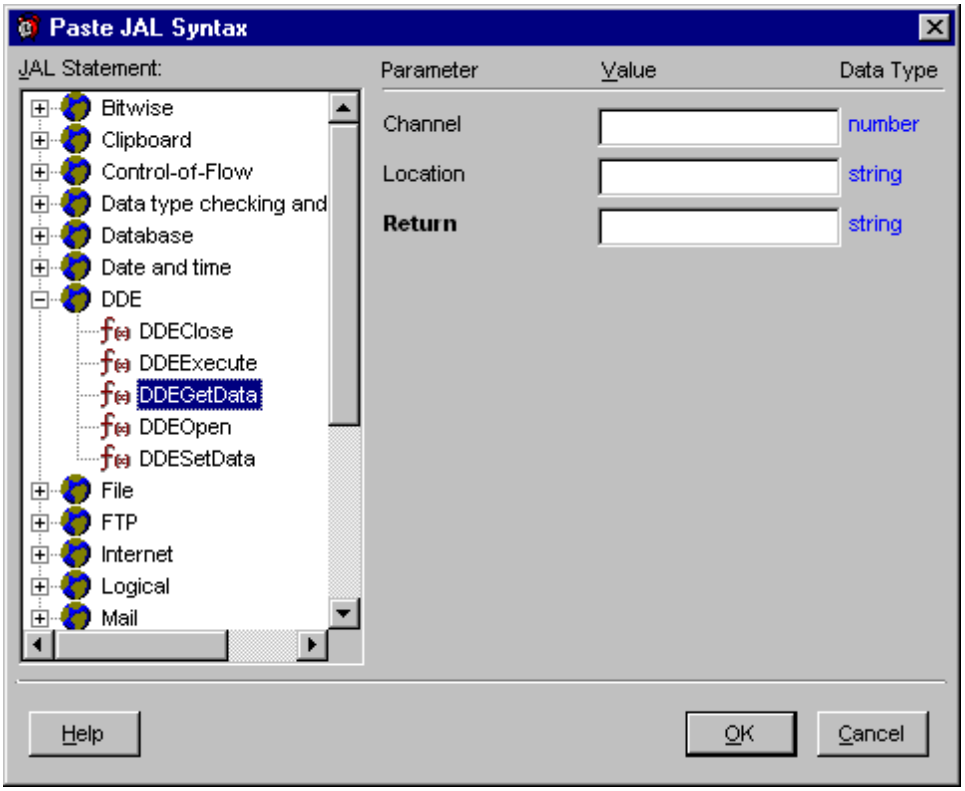
1 Click the  **Paste SQL** button on the Editor's toolbar. The Paste SQL Syntax dialog box will open.
2 Select the command that you want to paste from the drop-down list of available SQL commands. The syntax will appear in the Syntax box.
3 Click the **OK** button. The 24x7 Scheduler will paste the selected syntax into the script. You must complete the command by replacing the placeholders in the sample syntax and designating options, as applicable.

**Note:**
The 24x7 Scheduler provides syntax for commands available in most popular database systems. However, the list of valid SQL commands is not limited by the commands shown in the drop-down list. **You can use any valid SQL syntax supported by your database.**




### Pasting JAL Syntax

1 Click the  **Paste JAL** button on the Editor's toolbar. The Paste JAL Syntax dialog box will open.
2 Select the statement that you want to paste from the tree list of supported JAL statements.
3 Fill in, if applicable, the statement parameters displayed in the right pane of the dialog window.
4 Click the **OK** button. The 24x7 Scheduler will build the syntax and paste it into the script.




**Hot Keys**

| | |
|---|---|
| Mouse left button | Cursor movement |
| Mouse right button | Tracking pop-up menu |
| Arrows | Cursor movement |
| Ctrl+Left | One word left |
| Ctrl+Right | One word right |
| Ctrl+Up | To the beginning of the paragraph |
| Ctrl+Down | To the end of the paragraph |
| Home | To the beginning of the line |
| End | To the end of the line |
| PgDn | Page down |
| PgUp | Page up |
| Ctrl+Home | To the beginning of the text |
| Ctrl+End | To the end of the text |
| Ctrl+PgUp | To the beginning of the visible text |
| Ctrl+PgDown | To the end of the visible text |
| Shift | When combined with all previous keys (including mouse) expands current selection |
| Insert | Changes writing mode |
| Delete | Deletes one symbol (does not clear selection) |
| BkSp | Deletes one symbol (does not clear selection) |
| Ctrl+Y | Deletes one line (does not clear selection) |
| Ctrl+R | One word to the right |
| Ctrl+L | One word to the left |
| Ctrl+U | To the beginning of the visible text |
| Ctrl+D | To the end of the visible text |
| Ctrl+Z | Edit/Undo |
| Ctrl+Y | Edit/Redo |
| Ctrl+X | Edit/Cut |
| Ctrl+C | Edit/Copy |
| Ctrl+V | Edit/Paste |
| Ctrl+A | Edit/Select All |
| Ctrl+F | Edit/Find |

| | |
|---|---|
| Ctrl+H | Edit/Replace |
| Ctrl+G | Edit/Go to Line Number |
| Ctrl+Ins | Edit/Copy |
| Shift+Ins | Edit/Paste |
| Shift+Ctrl+Ins | Edit/Duplicate |
| Ctrl+Del | Edit/Clear |
| Shift+Del | Edit/Cut |
| Ctrl+BkSp | Edit/Undo |
| Shift+Ctrl+BkSp | Edit/Redo |
| F3 | Edit/Search Next |
| F1 | Help/Help Contents |
| Shift+F1 | Help/Help on Statement (on current word from caret position) |

# Chapter 14: Log Viewer

The Log Viewer is the tool you can use to monitor job events and view trace information, if tracing is enabled. With Log Viewer, you can also troubleshoot various job execution problems. The logging for the job execution is performed automatically, all other tracing information is collected according to the selected options in the 24x7 Scheduler preferences. For details, see **General** and **Network** tabs.

Start the Log Viewer by clicking the **View** menu then selecting **Log**, or simply by pressing the Ctrl L shortcut.

The Log Viewer consists of the six following tab pages:

- **Job Log** - displays the full event log stored in the SCHEDULE.LOG file for all scheduled jobs. See Execution Logs topic for details.
- **Master** - displays the tracing information stored in the MASTER.LOG file for the last Master session. See Fail-over Mode topic for details.
- **Standby** - displays the tracing information stored in the STANDBY.LOG file for the last Standby session. See Fail-over Mode topic for details.
- **24x7 Script** - displays the tracing information stored in the SCRIPT.LOG file for the last executed JAL script. See Job Automation Language topic for details.
- **Interface** - displays the tracing information stored in the INTFACE.LOG file for the last external interface session. See External Interface topic for details.
- **Database** - displays the tracing information stored in the PBTRACE.LOG file for the last database session. See Database Interfaces topic for details.
- **Statistics** - displays the job execution statistics log stored in the STAT.LOG file. This information is available on Windows NT (NT/2000/XP/2003/VISTA/2008/7) platform only. See Job Execution Statistics topic for details.

When you first open a log, the Log Viewer displays the current information for that log. That displayed log records are not refreshed while you are viewing. The view is refreshed the Log Viewer is reopened. The log is automatically updated only for the job event log.

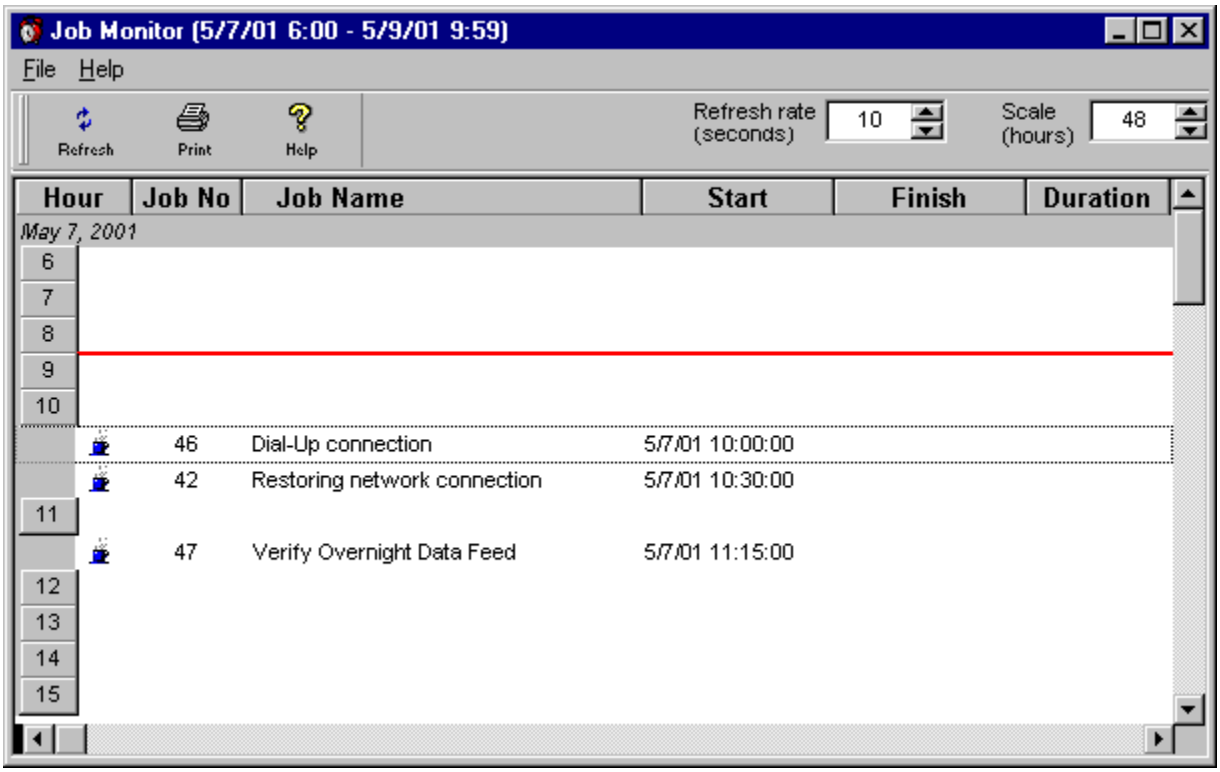
You should periodically review the main job event log to check for possible job execution problems. The depth of the job history in the log is limited by the **maximum number of entries in the log** parameter. You can change this parameter in the 24x7 Scheduler preferences. The 24x7 Scheduler, for performance reasons, keeps the log loaded in the computer memory. Therefore, the amount of memory required depends on how many records you have in that log. Under normal circumstances, you should let the 24x7 Scheduler to capture the job history for at least a week. This will allow you to view the historical status of your jobs. If, however, you created a job that runs frequently, such as once each minute, you should not allow large logs, unless you have set the **maximum number of entries in the log** parameter to a reasonable value.

 **Tips:**

- The Log Viewer displays the most recent log entries first. To see the oldest entries, scroll to the end or click anywhere inside the log text area, then press the Ctrl+End shortcut.
- If you would like to search for text in any particular log described above, you may want to open the log file using Windows Notepad then use the Search command.
- When the **Trace** option is enabled (see system preferences) the 24x7 Scheduler captures all trace information available. This process will slow the overall system performance, but will provide important information that may help you in troubleshooting scheduled jobs.

# Chapter 15: Job Monitoring

The 24x7 Scheduler includes a real-time monitor that you can use to monitor currently running jobs and produce a forecast of the processing scheduled over the next 24 hours. Use the **Time Scale** option if you would like to customize the length of the forecasted period.




By default, the view is refreshed every ten seconds. You can change the refresh rate by using the **Refresh Rate** field. Enter zero to disable the automatic refresh function.

**To open the Job Monitor:**

From the **Tools** menu, select the **Job Monitor** item or simply click 

button on the Job Explorer window toolbar.

**Note**:

- The Monitor cannot forecast jobs having non-time based schedules such as "file watch" , "process watch" , and "email watch", as it does not know when the specified job conditions will be met. However, you can use the **Job Forecast Report**, which provide enhanced job forecast including forecast for non-time based jobs.
- The Monitor can also run on the 24x7 Remote Agent. To start the monitor right-click on the agent's icon in the system tray and then chose Job Monitor item in the popup menu.
- The Monitor running on 24x7 servers (both agents and masters) also display active connections from other 24x7 remote components. Such connections appear in the monitor as jobs whose name consists of the remote user name follows by " - Remote Job". For example: *John Smith – Remote Job*. A unique number that is displayed in the Job ID column identifies each remote connection.
- The Monitor can be also used with **24x7 Remote Control** to monitor remote jobs being executed by 24x7 servers (e.g., **24x7 Remote Agents** and 24x7 Master Schedulers).
- Do not keep Job Monitor window always open because the monitor keeps accumulating data for the jobs that already ran, and this may slow down the job processing especially if you schedule many jobs. Job Monitor is an interactive tool to aid in job tracking and troubleshooting. To reset the Job Monitor, close and re-open it.

**See also:**

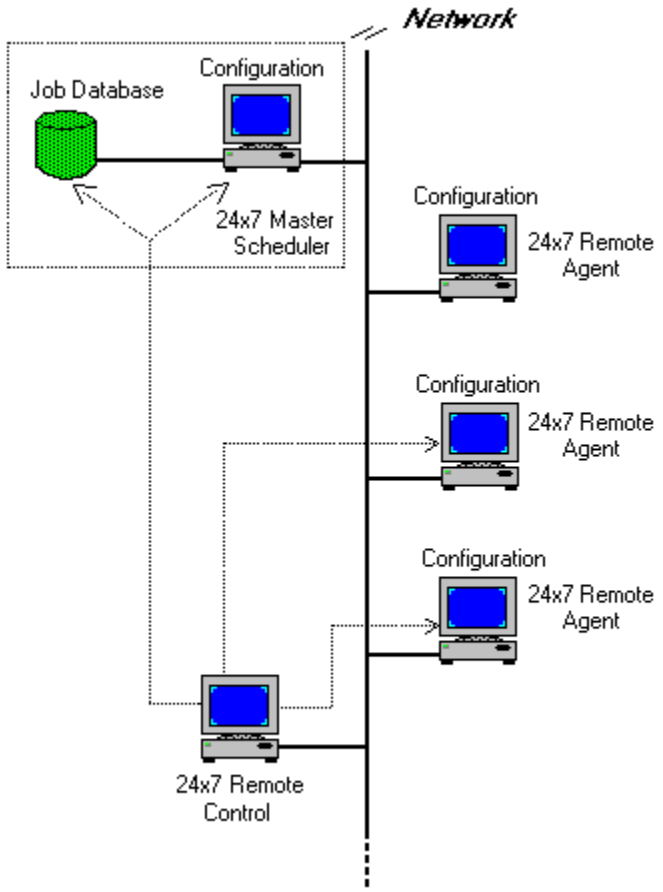
Status Report

Job Forecast Report

# Chapter 16: 24x7 Remote Control

## About 24x7 Remote Control

The 24x7 Scheduler software supports various configurations including networked components installed on remote computers and/or network, database and application servers that can be placed in locked rooms and closets. The 24x7 Remote Control allows developers, system and network administrators easily access such components without leaving their desks. This simply means that you can run the 24x7 Master Scheduler or 24x7 Remote Agent on one computer and access it from another computer connected to it via local or global network. This way you can manage jobs and configurations on remote computers without making a roundtrip to the remote computer. The 24x7 Remote Control utilizes built-in 24x7 Scheduler's distributed features eliminating the need to purchase and install third party remote access software.




The 24x7 Remote Control has been designed with management and security in mind. It simplifies enterprise job administration by making it easy for the administrator to configure all remote 24x7 components centrally from a **single** workstation. At the same time many people can **simultaneously** use 24x7 Remote Control to manage jobs on remote 24x7 servers. These features are due to 24x7 Scheduler's true client-server architecture.

**Important Notes:** Before accessing remote 24x7 Scheduler components, make sure they are running either in the **Master Scheduler** or **Remote Agent** mode. The 24x7 Remote Control can be used for manipulating remote or local 24x7 Scheduler servers only; it cannot be used for manipulating other software.

24x7 Remote Control supports the following functions:

- maintaining jobs and folders on remote 24x7 Master Schedulers,
- maintaining Script Libraries on remote 24x7 Master Schedulers and 24x7 Remote Agents,
- maintaining Holiday Lists on remote 24x7 Master Schedulers and 24x7 Remote Agents,
- maintaining Remote Agent profiles on remote 24x7 Master Schedulers and 24x7 Remote Agents,

- maintaining Database profiles on remote 24x7 Master Schedulers and 24x7 Remote Agents,
- downloading and displaying log files from remote 24x7 Master Schedulers and 24x7 Remote Agents.

**See also:**

About Remote Agents
Remote Agent Profiles
Managing Remote Jobs and Configurations

# Managing Remote Jobs and Configurations

You can use 24x7 Remote Control to control 24x7 Master Schedulers and 24x7 Remote Agents running on remote computers. The 24x7 Remote Control will work properly if all components are version 2.2.0 or later.

Before you connect to the remote host for the first time, you must create a new remote host profile. This profile will describe the connection parameters. You create remote host profiles in the same way that you create **remote agent profiles**.

**To add, modify, and delete profiles**:

1 Start 24x7 Remote Control.
2 Click the **File** menu, then click **Attach to Remote Host** menu item. The Remote Hosts Profiles dialog box will appear. This dialog box will also appear automatically when you start 24x7 Remote Control.
3 This dialog box is similar to the Remote Agent Profiles dialog box. See **remote agent profiles** help topic for instructions on how to create, delete or change profiles.

**To change definitions of remote jobs, view remote logs, and configure remote components:**

1 Select the **File/Attach to Remote Host** command from the menu. The Remote Hosts Profile dialog box will appear.
2 Select the desired profile then click the **Connect** button. The 24x7 Remote Control connects to the selected host, synchronizes with the remote job database and downloads remote logs and configuration for the connected component.
3 You can now make changes in the job and folder definitions, create new jobs and folders and delete those jobs not needed. You can also review remote logs using the **Log Viewer** and/or maintain definitions of **Holidays, Database Profiles**, and **Remote Agents** on the connected remote host.
4 You can change jobs, holidays, database profiles, and remote agent profiles the same way as you would if you were using the standalone version of the 24x7 Scheduler.
5 When the changes are completed, select the **File/Save to Remote Host** command. This will update the remote host. You can also click the **Save Remote** button on the Job Explorer toolbar.

 **Tip:** Keep in mind that 24x7 Master Scheduler is a true server application therefore many users can simultaneously attach and make changes in the Master Scheduler job database. If a job was changed or a new job was created after you attached to the 24x7 Master Scheduler you would not see the changes until you refreshed your local snapshot of the job database. To receive new changes made by other people, you will need to reattach to the remote host using the **File/Attach to Remote Host** menu.

**See also:**

About 24x7 Remote Control
Remote Agent Profiles
Starting 24x7 Remote Control
Maintenance for Holidays, Database Profiles, and Remote Agents

# Starting 24x7 Remote Control

**To start the 24x7 Remote Control from the command line:**

1 Change current directory to the 24x7 Scheduler directory. For example: *CD "C:\Program Files\24x7"*.
2 Run the command **24x7 /RCONTROL**. The 24x7 Remote Control will now start.

**To create a shortcut on the Desktop:**

You may prefer creating a shortcut to start the 24x7 Remote Control.

1 Right-click anywhere on the free area of the Desktop. A context menu will appear.
2 Click on **New**, then click **Shortcut**.
3 Type the command line for the 24x7 Scheduler ending with /RCONTROL parameter. The command line must include the full path to the **24x7.EXE**. For example *"C:\Program Files\24x7\24x7.EXE" /RCONTROL*.
4 Click the **Next** button.
5 Type the name for the newly created shortcut. For example: *24x7 Remote Control.*

Double-click on the shortcut icon to start the 24x7 Remote Control

**See also:**

About 24x7 Remote Control

# Chapter 17: 24x7 Scheduler API - External Interfaces

## External Interfaces Overview

The 24x7 Scheduler provides four different APIs for programmatic job control:

- **DOS command prompt interface** – this interface is based on the simple Job Definition Language (JDL). You can manipulate job definitions via this interface using the **24x7** command executed from the DOS prompt. This command requires JDL file name passed as one of the command line parameters. A JDL file is a flat ASCII text file that can be created manually in any text editor or be generated dynamically using virtually any programming language. For details, see Using JDL Files topic. The DOS command interface is appropriate for managing jobs 24x7 Scheduler running on the local computer.
- **DDE interface** – this interface is based on the standard Windows Dynamic Data Exchange (DDE) protocol. You can establish DDE links and call DDE functions from many programming environments such as MS Visual Basic, MS Access, C, PowerBuilder, Delphi, or any other language that can supports Windows DDE methods. For details, see Using DDE topic. The DDE interface is appropriate for managing jobs on 24x7 Scheduler running on the local computer.
- **COM+ interface** (called **24x7 Remote Control COM+** interface) – this interface can be used by any program that supports COM automation such as ASP, MS Visual Basic, MS Visual C++, MS Access, Delphi, PowerBuilder and many other. For details, see **24x7 Remote Control COM+ manual**. The COM+ interface is appropriate for managing jobs and controlling 24x7 Scheduler running on the local or remote computer. This interface requires that the 24x7 Scheduler must be running in a server mode: either as a **Master Scheduler** or **Remote Agent**.
- **Java interface** (called **24x7 Remote Control Java** interface) – this interface can be used by any Java program. Such program can run as standalone program, as an applet in a Web browser or as a servlet in Web server. For details, see **24x7 Remote Control Java manual**. The Java interface is appropriate for managing jobs and controlling 24x7 Scheduler running on the local or remote computer. This interface requires that the 24x7 Scheduler must be running in a server mode: either as a **Master Scheduler** or **Remote Agent**.

All external interfaces provide a way for programs to interact with the 24x7 Scheduler job engine in a consistent way.

## Using JDL Files

In addition to the advanced graphic user interface, 24x7 Scheduler supports simple DOS command line interface for accessing and manipulating job properties. You can use 24x7.exe command with the /SCRIPT option and a JDL file name as a parameter to execute JDL commands, for example, 24x7 /SCRIPT SUBMIT.JDL. The 24x7 Scheduler JDL files are regular text files that consist of JDL “commands", job “property names" and “values." A “property" must have an assigned “value" in the form property=value. A JDL file may also include comments. Comments may appear on a single line that starts with a semicolon (;). You can enter as many comment lines as you want. Blank lines may be used to improve readability. They do not need to be preceded by a semicolon. Spaces and tabs can also be used to improve the script structure.

 **Notes:**

- “Commands" and “property names" are case-insensitive; while “values" are case-sensitive.
- You can shortcut command names to three or more first characters, for example, **Delete**, **DELETE**, and **DEL**,. All three variants will delete the specified job.

You can use the Windows text editor to create and modify JDL files.

## JDL Commands

The following JDL commands are supported:

- **ADD –** Adds a new job.
  Format: ADD
  <property 1>=<value 1>

<property 2>=<value 2>
<property 3>=<value 3>
…
<property n>=<value n>

Parameters:

<property 1> .. <property n> - valid property name. See JDL Properties topic for the complete list of property names.

<value 1> .. /,value n> - new value for the specified property.

You may specify as many properties as you want. Although specifying one or more properties is optional, it is highly recommended that you specify a unique name for each new job.

 **Note:** A new job is created in the **Dynamic Jobs** folder. If this folder does not exist, it will be created automatically.

- **DELETE –** Deletes the specified job.
  Format: DELETE <job>
  Parameters:
  <job> - valid job ID or job name.

- **DISABLE –** Disables the specified job.
  Format: DISABLE <job>
  Parameters:
  <job> - valid job ID or job name.

- **ENABLE –** Enables the specified job.
  Format: ENABLE <job>
  Parameters:
  <job> - valid job ID or job name.

- **GET –** This command is available only via DDE interface.

- **SAVE –**Saves all changes in the job database.
  Format: SAVE
  Parameters:
  This command has no parameters.

- **SET –** Alters the specified property of the specified job.
  Format: SET <job> <property>=<value>
  Parameters:
  <job> - valid job ID or job name.
  <property> - valid property name. See JDL Properties topic for complete list of property names.
  <value> - new value for the specified job property.

The 24x7 Scheduler writes results of JDL file processing into JDL.LOG file in the 24x7 installation directory. You can use Windows Notepad or any other text viewer program to see contents of this file after processing. In addition, if the tracing features are enabled, the 24x7 Scheduler will log the internal JDL tracing information into the INTFACE.LOG file. You will be able to read this using the Log Viewer. This information can help in troubleshooting JDL processing issues.

**See also:**

JDL File Sample
JDL Overview
Using DDE
Job Properties

# Using Dynamic Data Exchange

You can use standard Windows Dynamic Data Exchange (DDE) protocol for programmatic job control from other applications. DDE provides a link for two Windows applications to communicate. Using DDE, one application (the client application), can request information from, or send commands to, another application (the server application). The server application then processes the request from the client application. The server performs a task such as updating data, or returning requested information to the client, such as an element of data maintained by the server application.

The 24x7 Scheduler is designed to act as a DDE server, with the ability to process DDE requests and commands from client applications.

To use DDE based JDL interface, your application must act as a DDE client. The following steps are required for typical DDE communication:

1 The client application establishes a "cold" DDE link to the 24x7 Scheduler by calling the corresponding DDE function. This function name varies in different programming languages. For example, in Visual Basic for Applications - it is DDEInitiate, in PowerBuilder - it is OpenChannel. You would use "24x7 Scheduler" for the DDE name and "JDL" for the DDE topic parameters required for establishing a DDE link to the 24x7 Scheduler.
2 The client application executes one or more DDE requests.
3 The client application terminates the DDE link by calling the corresponding DDE function. This function name varies in different programming languages. For example, in Visual Basic - it is DDETerminate, in PowerBuilder - it is CloseChannel.

For additional information on DDE, consult Microsoft's DDE documentation.

# JDL Commands

The following JDL commands are supported:

- **ADD –** Adds a new job.
  Format: ADD
  Method: DDE command
  Parameters:
  This command has no parameters.
  Return: None. You should immediately execute the **GET** command with the **NEW_ID** parameter to get the new job ID. You can call the **SET** command later to set/modify properties of the new job by using the returned job ID as a reference.
  Visual Basic example:

```
Application.DDEExecute ChannelNumber, "ADD" 'create new blank job
New_Job = Application.DDERequest ChannelNumber, "New_ID" 'retrieve ID of newly created job
```

  **Notes:**
  A new job is created in the **Dynamic Jobs** folder. If this folder does not exist, it will be created automatically.

- **DELETE –** Deletes the specified job.
  Format: DELETE <job>
  Method: DDE command
  Parameters:
  <job> - valid job ID or job name.
  Visual Basic example:

```
Application.DDEExecute ChannelNumber, "DELETE 5"'delete job #5
```

- **DISABLE –** Disables the specified job.
  Format: DISABLE <job>
  Method: DDE command
  Parameters:
  <job> - valid job ID or job name.
  Visual Basic example:

```
Application.DDEExecute ChannelNumber, "DISABLE 5"'disable job #5
```

- **ENABLE –** Enables the specified job.
  Format: ENABLE <job>
  Method: DDE command
  Parameters:
  <job> - valid job ID or job name.
  Visual Basic example:
  Application.DDEExecute ChannelNumber, "ENABLE 5"'enable job #5

- **GET –** Retrieves the specified property of the specified job.
  Format: GET <job><tab><property>
  Method: DDE request data
  Parameters:
  <job> - valid job ID or job name.
  <tab> - tab character (ASCII code 9)
  <property> - valid property name. See JDL Properties topic for complete list of property names.
  Return: value for the specified property
  Visual Basic example:
  Start_Time = Application.DDERequest ChannelNumber, "5"+ Chr(9) + "Start_Time" 'retrieve start time for job #5

- **SAVE –**Saves all changes in the job database.
  Format: SAVE
  Method: DDE command
  Parameters:
  This command has no parameters.
  Visual Basic example:
  Application.DDEExecute ChannelNumber, "SAVE" 'save changes in the job database

- **SET –** Alters the specified property of the specified job.
  Format: SET <job><tab><property> <value>
  Method: DDE send data
  Parameters:
  <job> - valid job ID or job name.
  <tab> - tab character (ASCII code 9)
  <property> - valid property name. See JDL Properties topic for complete list of property names.
  <value> - new value for the specified job property.
  Visual Basic example:
  Application.DDEPoke ChannelNumber, "5"+ Chr(9) + "Start_Time" , "9:30"'change start time for job #5

If tracing features are enabled, the 24x7 Scheduler will log the internal DDE tracing information into the INTFACE.LOG file. You can view this information using the Log Viewer. This information can help when troubleshooting the JDL interface

**Note:**
Before you establish a DDE link, make sure that the 24x7 Scheduler is running.

**See also:**
JDL Interface Examples
JDL Overview
Using JDL Files
Job Properties

# JDL Properties

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

| Name | Meaning |
|---|---|
| **ACCOUNT** | E-mail Account such as user ID, profile, or e-mail address (e-mail watch job).<br>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. |
| **ALL_DAY_TYPE** | All Day Schedule Type, one of the following: **R, L** (**R** - recursive, repeat at specified intervals; **L** - fixed time list) |
| **AGENT** | Same as Host (see Host description) |
| **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) |
| **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) |
| **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 :<br>[old password][tab character][new password][tab character][protection state]<br>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 |
| **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 |
| **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 |
| **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 |
| **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).<br>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.<br>Example map:<br>SE,FE,EE,FD |

| | |
|---|---|
| | This example map represents the following Notification Events and Events:<br>1. Send notification email on job start.<br>2. Send notification email on job finish.<br>3. Send notification email on job error.<br>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) |
| **MSG_NET_RECIPIENT** | Message Recipient for Notification Action of Network Message Type |
| **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) |
| **MSG_PAGER** | Page Recipient's Pager Number |
| **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 |
| **MSG_SCRIPT_TYPE** | Type of Script for Notification Action of Script Type, one of the following: **JAL**, **VBS** (Job Automation Language, Visual Basic Script) |
| **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 |
| **NUMBER_OF_RETRIES** | Maximum Number of Retries Before Job Gets Marked as Failed (number) |
| **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) |
| **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. |
| **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** | JAL Script |
| **SCRIPT_TYPE** | Job Script Type, one of the following: **JAL**, **VBS** (Job Automation Language, Visual Basic Script) |
| **SEND_KEYSTROKE** | Send Keystroke, one of the following: **Y**, **N** (yes, no) |
| **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) |

For information on properties description and usage see Job Properties topic.

**See also:**

JDL Interface Examples
JDL Overview
Using JDL Files
Using DDE

# Examples of Using Job Definition Language Commands

**1. Command Files**

The following example demonstrates how to use JDL command line interface. This example includes two files: EXAMPLE.JDL and JDL.BAT. Note that the JDL.BAT file is generic.

**JDL.BAT**

```
@echo off
echo the 24x7 Scheduler
echo Processing JDL command file %1%
24x7 /SCRIPT %1
type jdl.log
```

**EXAMPLE.JDL**

```
; Change name for the job #1
SET 1 NAME=JDL command file test 1
; Change start-in directory for the job #2
SET 2 STARTIN=c:\windows
; Add the new job to the schedule
ADD
        NAME=New job submitted via JDL command file
        COMMAND=notepad
        ASYNC=N
        TIMEOUT=1
        DESCRIPTION=This job was created using JDL command file
; Save all changes
SAV
```

To try this example:

1 Create both required files.
2 Make sure the 24x7 Scheduler is running.
3 In the DOS prompt run JDL EXAMPLE.JDL.

**2. DDE interface**

This example shows how to use Visual Basic to execute JDL commands using DDE protocol.

```
Attribute VB_Name = "24x7"
Option Compare Database
Option Explicit

Sub Macro_24x7( )
'
' Establish DDE link to the 24x7 Scheduler and send some JDL commands
'
'
    Dim ChannelNumber As Long, NewJob As Long, JobName As String


    ' initiate DDE conversation
    ChannelNumber = Application.DDEInitiate("24x7 Scheduler" , "JDL" )

    ' get job #1 name
    JobName = DDEGetValue(ChannelNumber, "1"+ Chr(9) + "NAME" )

    ' disable job #1
    Call DDEDisable(ChannelNumber, "1")
    ' or alternatively perform the same operation using job name
    ' instead of using job number. This works for all commands
    ' Call DDEDisable(ChannelNumber, JobName)

    ' change job #1 name
    Call DDESetValue(ChannelNumber, "1"+ Chr(9) + "Name" , "New Name for job #1")

    ' delete job #1
    Call DDEDelete(ChannelNumber, "1")

    ' add new job
    NewJob = DDEAdd(ChannelNumber)
    If NewJob > 0 Then ' set new job properties
        Call DDESetValue(ChannelNumber, CStr(NewJob) + Chr(9) + "Name" , "New Name"
)
        Call DDESetValue(ChannelNumber, CStr(NewJob) + Chr(9) + "Command" ,
"c:\feed\mfeed.ex /S" )
        Call DDESetValue(ChannelNumber, CStr(NewJob) + Chr(9) + "Start_Time" ,
"19:30")
        Call DDESetValue(ChannelNumber, CStr(NewJob) + Chr(9) + "Async" , "N" )
        Call DDESetValue(ChannelNumber, CStr(NewJob) + Chr(9) + "Timeout" , "30")
    End If

    ' save all changes
    'Call DDESave(ChannelNumber)


    ' terminate DDE conversation
    Application.DDETerminate ChannelNumber
End Sub

Function DDEGetValue(ChannelNumber As Long, Location As String) As String
    DDEGetValue = Application.DDERequest(ChannelNumber, Location)
End Function

Sub DDESetValue(ChannelNumber As Long, Location As String, Data As String)
    Application.DDEPoke ChannelNumber, Location, Data
End Sub

Sub DDEDisable(ChannelNumber As Long, Location As String)
    Application.DDEExecute ChannelNumber, "DIS "+ Location
End Sub
```

```
Sub DDEEnable(ChannelNumber As Long, Location As String)
    Application.DDEExecute ChannelNumber, "ENA "+ Location
End Sub

Sub DDEDelete(ChannelNumber As Long, Location As String)
    Application.DDEExecute ChannelNumber, "DEL "+ Location
End Sub

Sub DDESave(ChannelNumber As Long)
    Application.DDEExecute ChannelNumber, "SAV"
End Sub


Function DDEAdd(ChannelNumber As Long) As Long
    Application.DDEExecute ChannelNumber, "ADD"
    DDEAdd = DDEGetValue(ChannelNumber, "NEW_ID" )
End Function
```

To try this example:

1 Copy the code to the application supporting Visual Basic for Applications programming language.
2 Make sure the 24x7 Scheduler is running.
3 Run the Macro_24x7 macro.

 **Important note:**

*The above example will not work in Microsoft Excel due to Excel specifics*

# Chapter 18: System Options

The 24x7 Scheduler can be reconfigured to some extent. Select the **Tools/Options** menu to open the Options dialog. You can customize the following features:

## General Options

- **Maximum error message display time** – This parameter sets the maximum display time for an error message box before the message disappears. To disable appearance of error messages, set this parameter to zero.
- **Ask confirmation on exit** – This parameter controls appearance of the Exit confirmation message. This option is highly recommended because the prompt message may include information on any active jobs and connections at the time of the exit. Exiting the system while jobs are running or an active database operation is in progress, can cause unpredictable results. You should disable this prompt only in very special circumstances such as when a user input is not possible or undesirable. Note that rebooting the computer using either the 24x7 Scheduler's "Reboot after job" feature or the Reboot statement does not produce the confirmation prompt.
- **Automatically save changes on exit** – This allows 24x7 Scheduler to save any pending changes in the job database without displaying confirming dialog.
- **Reset job queue on startup** – This parameter controls how 24x7 Scheduler processes pending jobs left in job queues since last scheduler shutdown. If this option is enabled 24x7 Scheduler deletes old pending jobs and gets a fresh start

## Email and Pager Options

For detailed information about supported email interfaces see **Sending and Receiving Email Messages** topic.

- **Email interface** – This parameter controls which email interface 24x7 Scheduler uses for sending various notification messages and executing Job Automation Language mail statements.
- **Send method** – This parameter is used with MAPI email interface. Different email programs use different Windows MAPI properties. This parameter allows you to select send method compatible with your MAPI email client software such as MS Outlook, Netscape, Eudora and other.
- **Message encoding** – This parameter is used with SMTP email interface. Different email programs use different encoding. Use this parameter to configure encoding method compatible with the email program used by the message recipients.
- **Message content type** – This parameter is used with SMTP email interface. It controls format in which SMTP email messages are generated.
- **SMTP email server** – This parameter is used with SMTP email interface. Use it to specify name or IP address of your SMTP server.
- **SMTP sender address** – This parameter is used with SMTP email interface. Use it to specify default SMTP message originator address. By default this parameter is set to *24x7_Scheduler@24x7automation.com* . Change this parameter to a valid email address if your SMTP email server requires authentication.
- **SNPP server** – This parameter is the name or IP address or your paging server. Enter value provided by your paging service carrier.

# Fail-over Mode and Distributed Service Options

You use this set of options to configure 24x7 Scheduler distributed features for the Master and Standby scheduler, and the Remote Agents. For more information on **Fail-over mode** and connection parameters see About Fail-Over Mode topic.

- **Multi-instance synchronization enabled** – Check this option to enable Master/Standby scheduler synchronization.
- **Synchronization interval** - Specifies the interval (in seconds) at which you want the Standby scheduler to connect to the Master schedule and synchronize their job databases.
- **Number of failed synchronizations –** Specifies how many sequential failed synchronization attempts indicate that the Master scheduler is not available and the Standby scheduler should take over.

# Editor Options

- **Font** - Clicking on the **Font** button opens the Select Font dialog box. Use this option to customize font characteristics for the JAL, VBS and SQL Editors.
- **Automatically save a script recovery file** – This allows periodic saving of the backup copy of the script opened in the 24x7 Script Editor. You may want to use this option while you work on a script and you have 24x7 Scheduler running other jobs in background. If your computer hangs (stops responding) or you lose power unexpectedly, you can use the **AutoSave.jdl** file to recover your script the next time you start 24x7 Editor.
- **Autosave interval** – This specifies the interval (in minutes) at which you want the Editor to update the recovery file.
- **Search for special ASCII characters** – This allows 24x7 Scheduler to search for special ASCII characters escape symbols and substitute them with the appropriate ASCII characters.
- **Escape character** – This designates the character to be used as a prefix for special ASCII characters. Backslash (\) is the default escape character.

 **Note:**
IMPORTANT: AutoSave does not replace the Save command. You must still save your job definition and script changes when you finish working on them.

# Log and Debug Options

- **Load log on startup** – This instructs the 24x7 Scheduler to load the job execution log on startup. Clearing this check box will disable log loading and reduce 24x7 Scheduler loading time. However, you will not be able to see old log entries in the Job Explorer.
- **Maximum number of entries in the log file** – The depth of the job history in the log is limited by the **maximum number of entries in the log** parameter. The 24x7 Scheduler, for performance reasons, keeps the log loaded in the computer memory. Therefore, the amount of memory required depends on how many records you have in that log. Under normal circumstances, you should let the 24x7 Scheduler capture the job history for at least a week. This will allow you to view the historical status of your jobs. If however, you created a job that runs frequently, such as once each minute, you should not allow large logs and set the **maximum number of entries in the log** parameter to a reasonable value.
- **Clear log on startup** – This instructs the 24x7 Scheduler to empty job execution logs on startup. Clear this check box to disable clearing of the job history. Under normal circumstances, you should let the 24x7 Scheduler capture the job history for at least a week. However, clearing large logs will save some disk space.

- **Generate status report (HTML)** – This allows updating of the Status Report after each job run. The 24x7 Scheduler is capable of duplicating job execution log entries in an elegant HTML report format and automatically updating this report each time a new entry is added to the log file. This report can reside on your company Web server, allowing you to view the Status Report using any Web browser that supports frames. You will therefore be able to monitor job execution over the Internet.

- **Status Report directory** – This specifies the destination directory of the Status Report. Alternatively, you can click the **Browse** button to select the desired directory. If you leave the field blank, the 24x7 Scheduler will save reports in the installation directory. If your company Web server can be accessed via Intranet, you can specify the destination directory on the Web server to have the 24x7 Scheduler automatically upload the Status Report.

In addition, you can set the following properties used to troubleshoot job execution and communications between 24x7 Scheduler components.

- **Trace enabled** – Enables tracing. This option is used to:
  1 Troubleshoot Standby/Master Scheduler connections. When tracing is enabled, 24x7 Scheduler traces all activity between the Standby and the Master components. This internal information can be helpful when debugging, including memory usage, an internal call trace, and the types and values for passed parameters.
  2 Troubleshoot connections to Remote Agents. When tracing is enabled, 24x7 Scheduler traces all activity between the main scheduler and the Remote Agent. This internal information can be helpful when debugging, including memory usage, an internal call trace, and the types and values for passed parameters.
  3 Debug JAL script execution. When tracing is enabled, 24x7 Scheduler traces all executed script statements including statement parameters and returned values. This internal information can be helpful when debugging JAL scripts. Some JAL statements also output additional information that be visible only in the trace.
  4 Debug Telnet operations. The tracing data is saved in Telnet.log file
  5 Debug external interfaces, both command line and DDE interfaces. When tracing is enabled, 24x7 Scheduler traces all received commands including command parameters and internal information on command processing. This information can be helpful when troubleshooting external interfaces.

- **Database trace enabled** – Enables database tracing. This option is used to record the internal commands performed by 24x7 Scheduler while communicating with a database. This internal information can be helpful when troubleshooting database operations.

- **Job execution statistics enabled** (Windows NT/2000/XP/2003/VISTA/2008/7 platforms only) – Enables collection of Job Execution statistics. You can use collected execution statistics to better understand the behavior of scheduled programs, such as processors and memory usage, run-time duration and delays. This information may also help you when troubleshooting job anomalies.

- **Logging on to the system event log enabled** (Windows NT/2000/XP/2003/VISTA/2008/7 platforms only) – Enables parallel logging of all job log entries in the Windows NT Application Log. For detail on the Windows NT Application Log, click **Start** button, then choose **Help** command from the Windows NT Start menu.

- **Job execution status display enabled** - Enables displaying of the status/progress message box while executing scheduled jobs. The status/progress message box allows you to watch job execution progress.

## Service Options for Windows NT/2000/XP/2003/VISTA/2008/7

This set of options is available on Windows NT (NT/2000/XP/2003/VISTA/2008/7) platforms only

- **Run 24x7 as a Windows NT service** – Check this option to install the 24x7 schedule service. Uncheck this options to uninstall the 24x7 schedule service.
- **Service account name** – Name of Windows NT account for which the 24x7 schedule service will be installed. This must be either the system account specified as "LocalSystem" or a user account specified using ".\<Local User>"format (where <Local user> must be substituted with actual account name) or "<Domain>\<User>"format.
- **Service start options** – Enables/disable the 24x7 schedule service to start automatically when the specified user logs on. When local system account is selected and "Start automatically" options is chosen, the 24x7 Scheduler starts automatically whenever the computer is started and runs continuously in the background, regardless of whether a user is logged on.

- **Service mode** – Specifies whether the 24x7 schedule service functions as a standalone 24x7 Scheduler or as a 24x7 Remote Agent.

**Notes:**

- You can also modify parameters of an existing 24x7 schedule service by using the Services Control Panel Applet.
- You must have system administrator privileges in order to install, uninstall, or modify the 24x7 schedule service.
- The 24x7 scheduling service can be also installed and uninstalled using the 24x7 setup program. See **Silent Setup** topic for more details.

## Service Options for Windows 95/98/Me

This set of options is available on Windows 95, Windows 98 and Windows Me platforms only

- **Run 24x7 as a Windows 95/98/Me service** – Check this option to install the 24x7 schedule service. Uncheck this options to uninstall the 24x7 schedule service. When the service is installed, 24x7 Scheduler starts automatically whenever the computer is started.
- **Survive user log off** – Check this option if you do not want 24x7 to exit on "Close all programs and log on as a different user" Windows operation
- **Service start options** – Always "Start automatically".
- **Service mode** – Specifies whether the 24x7 schedule service functions as a standalone 24x7 Scheduler or as a 24x7 Remote Agent.

## Other Options

- **Monitor free system resources** – This option enables 24x7 to monitor and display free system resources. The resource meter is displayed in the **Status Bar**. When the resource meter is enabled 24x7 tracks amount of the free system memory and whenever it falls below certain thresholds, it writes warning messages to the main job log file. When the resource miter is disabled 24x7 does not display the resource meter and does not track amount of the free system memory.
- **Automatically restart computer when free memory drops below 10%** – This option allows 24x7 to restart the system in an emergency case such as when the amount of free system memory drops below 10% of the total memory available on the system.
- **Enable periodic restarts** – Use this option to schedule periodic 24x7 Scheduler or 24x7 Remote Agent restarts. This way you can deal with jobs that cause various resource leaks. Using this option you can recycle the scheduler/agent and this way recover lost resources.
- **Show tip of the day on 24x7 Scheduler startup** – This option controls whether 24x7 Scheduler displays **Tip Of The Day** message on startup.
- **Show job save reminder** – This option controls whether 24x7 Scheduler displays Save Job reminder whenever a new job is created or an existing job is modified using **Job Properties Wizard**.

# Chapter 19: 24x7 Job Automation Language

## Overview

You can write job automation scripts using Job Automation Language (JAL), the 24x7 Scheduler primary scripting language. JAL provides you with great flexibility in scheduling and executing different kinds of jobs. It allows you to build complex job dependencies; automate and control the execution of other programs with ability to simulate an operator by sending various keystrokes to the program being executed as well as perform advanced error checking and handling. JAL includes over 300 statements for a wide variety of operations such as file manipulations, program execution, process handling, database operations, and many other.

As with other parts of the 24x7 Scheduler, you can use macro-parameters in your JAL and VB scripts. Macro-parameter values are substituted in job run-time just before the JAL interpreter starts script execution.

The 24x7 Scheduler provides you with a built-in JAL editor that you can use to create new, as well as modify existing, JAL, VBS and SQL scripts.

**See also:**

Syntax

Statements by Category

## Job Scripts vs. Script Files

It is import to distinguish job scripts from script files. Job scripts are stored encrypted and secure within the job database file and executed by the built-in JAL script engine during job run time. You do not need to manage them separately from the schedule. You do not need to manually deploy them to remote agents. You do not need to worry about saving user names, password and other sensitive information in files. 24x7 Scheduler takes care of all of these issues. 24x7 Scheduler also provides built-in script editors and debuggers.

In comparison, scripts saved in external files are executed by the scheduler as external programs, in other words as command line jobs. If you need to run such jobs remotely, you must take care of deploying such scripts to remote systems. However, one of the advantages of external scripts is that they can be run independently from 24x7 Scheduler.

To run external VBScript files you can use Windows' built in **cscript.exe** command. To run external JAL script files you can use **jalscript.exe** command installed with 24x7 Automation Suite software. You can also start .JAL files by simple entering their names at the DOS command prompt.

**See also:**

Syntax

Statements by Category

## Syntax

The 24x7 Job Automation Language supports the following three programming language elements:

1 Variables
2 Statements
3 Comments

# Variables

A JAL variable is a named storage location that contains data that can be modified during program execution. Each variable has a unique name that identifies it within the JAL script. You must declare a variable before you can use it. Use the Dim statement to declare variables and their data types. When you declare a variable, you can accept the default initial value or specify an initial value.

The variable name should begin with an alphabetic character, followed by any combination of alphabetic characters, numbers, or the underscore character, as in this example: ProcessID.

A variable can be declared as one of the following data types:

- NUMBER - A signed floating-point number with 15 digits of precision and a range from 2.2250738585072E-308 to 1.79769313486232E+308. When necessary, the 24x7 Scheduler automatically handles conversion between floating-point and integer numbers. Variables of NUMBER data type have 0 (zero) as their default initial value.
- STRING - Any string of ASCII characters with variable length (0 to 2,147483,647). Variables of STRING data type have ""(empty string) as their default initial value.
- DATE - The date, including the full year (1000 to 3000), the number of the month (01 to 12), and the day (01 to 31). Variables of DATE data type have 1900-01-01 (January 1, 1900) as their default initial value.
- TIME - The time in 24-hour format, including the hour (00 to 23), minute (00 to 59), and second (00 to 59), with a range from 00:00:00 to 23:59:59. Variables of TIME data type have 00:00:00 (midnight) as their default initial value.
- DATETIME - The date and time in a single data type. Variables of DATETIME data type have 1900-01-01 00:00:00 (January 1, 1900 midnight) as their default initial value.
- BOOLEAN - Contains TRUE or FALSE. Variables of BOOLEAN data type have FALSE as their default initial value.

A variable can have either local or global scope. A local variable is a temporary variable accessible only from within the script in which you define it. When the script is finished, the variable constant ceases to exist. Global variables are accessible from anywhere in any script of any job. To avoid ambiguity when referring to variables, you must use the GLOBAL prefix when declaring or referring to global variables. The 24x7 Scheduler allocates memory and initializes a global variable whenever it executes a JAL script, in which it first time finds the declaration of that variable. 24x7 Scheduler ignores the declaration on subsequent runs. You can use the Set statement to change the values of global variables.

Examples:
This statement declares a local variable: Dim( counter, number )
This statement declares a global variable: Dim( global.counter, number )
This statement sets the value of a local variable: Set( count, 5 )
This statement sets the value of a global variable: Set( global.count, 5 )

# Statements

JAL statements have two main components: the statement name and the statement parameters. The statement and variable names are case insensitive. The statement name must be spelled exactly as it is shown in the syntax and the parameters must be used in the order they are given in the syntax. Parameters provide information for the statement. For example, the FileReadLine statement, which reads a line from a text file, has parameters for the name of the file, the line number, and the variable name into which the line will be stored.

All JAL statements use the following syntax:
StatementName(parameter1, parameter2, ...)

The opening parenthesis, closing parenthesis, and commas are optional. You can use spaces and tabs instead. Statement names are not case-sensitive. The 24x7 Scheduler automatically removes all unnecessary white space before executing JAL script. This enables you to use spaces and tabs in the script to improve readability.

Parameters can be variable names, text strings or numbers, and are separated by commas, spaces, and/or tabs. Text string parameters must be enclosed in double quotes. The number of white space characters is preserved when they are part of a string literal (parameter).

Some statements have return value. The return value is always passed in the last statement parameter. When coding statements you should declare a variable of the appropriate type in order to store this return value.

Some statements have no parameters at all. For example: Exit
You cannot nest more than one statement in a statement string.

@ sign is used as an escape character that allows you to include **macro-variables** as part of a statement parameter. If you need a literal @ sign (for example, in a string specification) you must double the @ sign.

You can define your own JAL statements. You can use the **Script Library** tool (Tools/Script Library menu) to create, modify, and delete your **user-defined JAL statements**. User-defined JAL statements may call other user-defined statements. You should be very careful when coding your JAL statements and avoid cyclical calls that may lead to infinite loops.

# Statement continuation

Although typically you would put one statement on each line, you will occasionally want to continue a statement on to more than one line. The statement continuation character is the ampersand (&).

Syntax:

*Start of statement &*
*more statement &*
*end of statement*

The ampersand must be the last nonwhite character on the line (or the script interpreter will consider it as a part of the statement).

# Special ASCII characters

You can include special ASCII characters in strings. For example, you may want to include a tab in a string to ensure proper spacing or a new line character to indicate end of line. The backslash character (\) introduces special characters. To specify the backslash character as string literal, precede it with another backslash. Instead of the backslash character (\) you can specify another character to be used as a prefix for special characters. See 24x7 Scheduler Tools/Options for details.

Syntax:

| ***Symbol*** | ***ASCII character*** |
|---|---|
| *\n* | *New line* |
| *\r* | *Carriage return* |
| *\t* | *Tab* |
| *\"* | *Double quote* |
| ** | *Backslash* |

# Off-line scripts

You can develop off-line JAL scripts that are saved in text files as opposed to in-line scripts that are stored in the job database. You use the @SCRIPT tag in in-line scripts to reference off-line scripts. In run-time, 24x7 Scheduler inserts the contents of the off-line script file into the body of the main job script.

Syntax:
*@SCRIPT:<file name>*

For example: @SCRIPT:c:\scripts\check_errors.jal

# Comments

A comment is a line that starts with // (double forward slash). You cannot use comments on the line that already has some statement or statement. You can have as many comments as you want.

**Important Notes:**

- JAL script lines cannot exceed 8K (8192 characters) in size.
- The total script size cannot exceed 32000 characters.
- Nested statements are not currently supported.

**See also:**

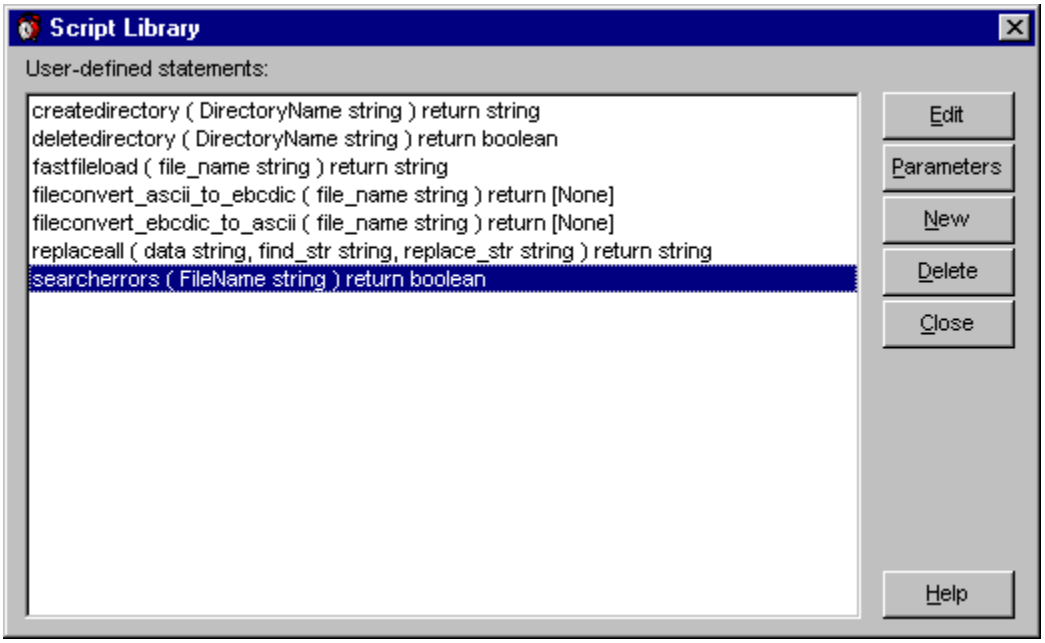
Statements by Category

Functions in Filter and Sort Expressions

# Script Library

You can access the Script Library from the **Tools/Script Library** menu or simply by pressing the F7 hot key.

The Script Library is the place where you can create and store user-defined JAL statements. User-defined JAL statements have global scope and can be called from any script type job. You call them from JAL scripts just as you call built-in JAL statement. From a user-defined JAL statement you can also call other user-defined statements and jobs. To simplify and easy job maintenance, implement common business functions as user-defined statements that you can call from multiple jobs.




User-defined statements must follow the same syntax rules that apply to other JAL scripts. Just as script jobs, variables declared in any user-defined JAL statement have local scope unless they are declared as GLOBAL. User-defined statements, just like built-in JAL statements, may have parameters. Parameters are defined at the time when you declare new user-defined statement in the Script Library. Every parameter must have a name. Parameters are always passed by value and inside the user-defined statement they are treated as local variables. In the statement code, you can refer to them by their names. However, their names are not used when calling the statement from

some other script. The position of the parameter in the user-defined statement is important. The order in which you specify parameter is the order you will use when calling the statement.

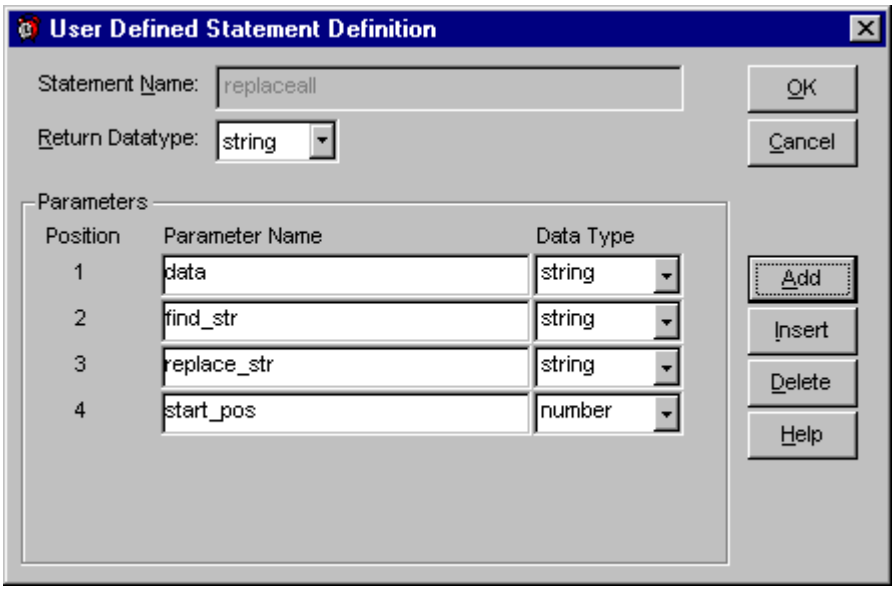
If you declare your user-defined statement with the return value, you must use the Return statement to return some value to the calling script. To call such user-defined statement, specify appropriate variable for the return value as the last parameter in the statement call line.
For example: *MyStatement first_param_as_var, "abc", 123, "third param as string", ret_value_var*.

**Caution:** User-defined statements can be recursive; that is, they can call themselves to perform a given task. However, recursion can lead to stack overflow. Be very careful when implementing recursive calls.

JAL is not case sensitive, that is, when calling your user-defined statements you can specify their names in lower, upper, or mixed case.

**To create a new user-defined statement:**

1 Open **Script Library** from the **Tools** menu. The Script Library dialog box will appear.
2 Click the **New** button. The Statement Definition dialog box will appear.
3 Specify the new statement name. The name must follow standard naming convention and must not exceed 50 characters in length. Generally, you should not use names that are the same as the names of the built-in JAL statements. Otherwise, you would shadow the same keywords in the language.
4 Specify the return value type or leave the return value (None) if the new statement has no return value.
5 Specify the statement parameter names and their data types. Parameter names must follow standard naming convention and must not exceed 50 characters in length. You can specify as many parameters as needed. To add more parameters, use the **Add** button.
6 When you are finished defining the parameters, click the **OK** button. The Script Editor window will appear.
7 Code your user-defined statement then click the **Exit** button to close the Script Editor and return to the Script Library.




**To modify an existing user-defined statement:**

1 Open the **Script Library** from the **Tools** menu. The Script Library dialog box will appear.
2 Find and select the desired statement.
3 To modify statement parameters, click the **Parameter** button. The Statement Definition dialog box will appear.
4 To modify the statement code, click the **Edit** button. The Script Editor window will appear.
5 Code your user-defined statement then click the **Exit** button to close the Script Editor and return to the Script Library.

**To delete an existing user-defined statement:**

1 Open the **Script Library** from the **Tools** menu. The Script Library dialog box will appear.
2 Find and select the desired statement.
3 Click the **Delete** button.

# Control-of-Flow Statements

24x7 Job Automation Language includes the following statements for controlling script execution logic, declaring variables, and assigning values to variables:

## Break

**Description:** In a LoopWhile, LoopUntil, or a ForNext control structure, passes control out of the current loop. Break takes no arguments.

**Syntax:** Break

**Usage:** A Break statement in a loop control structure causes control to pass to the statement following the end of loop label. In a nested loop, a Break statement passes control out of the current loop structure.
For information on how to jump to the end of the loop and continue looping, see Continue statement.

## Continue

**Description:** In a LoopWhile, LoopUntil, or a ForNext control structure, skips statements in the current loop. Continue takes no arguments.

**Syntax:** Continue

**Usage:** When 24x7 Scheduler encounters a Continue statement in a loop structure, control passes to the loop's end label. The statements between the Continue statement and the loop's end label are skipped in the current iteration of the loop. In a nested loop, a Continue statement bypasses statements in the current loop structure.
For information on how to break out of the loop, see Breakstatement.

## Dim

**Description:** Declares variables and allocates storage space.

**Syntax:** Dim variable, datatype [, initial_value]

| Argument | Description |
|---|---|
| variable | The name of the variable you want to declare |
| datatype | The data type of the variable. The following data types are supported: |

| | |
|---|---|
| | • String<br>• Number<br>• Boolean<br>• Date<br>• Time<br>• DateTime |
| initial_value | The initial value of the variable. The data type of the initial_value must match the declared data type of the variable. This argument is optional. |

**Usage Notes:** To declare a global variable, use *dot notation* with the GLOBAL prefix in front of the variable name.

## Exit

**Description:** Immediately exits the script.

**Syntax:** Exit

**Usage**: This statement has no arguments.

## RaiseError

**Description:** Causes an error event at the job level and writes error message to the 24x7 event log.

**Syntax:** RaiseError error

| Argument | Description |
|---|---|
| error | A string whose value is the error message that you want to write to the 24x7 event log. |

**Usage**: By default, RaiseError statement halts job execution and writes error message to the 24x7 event log. If parallel logging to the Windows NT (NT/2000/XP/2003/VISTA/2008/7) event log enabled, RaiseError also writes error message to the Windows NT (NT/2000/XP/2003/VISTA/2008/7) application event log. The exact behavior of the RaiseError statement depends on the error mode set by any of the previously executed OnError statements.

**See also:**

LogAddMessageEx
Exit
OnErrorStop
OnErrorResumeNext
OnErrorGoTo

# ForNext

**Description:** A control structure that is a numerical iteration, used to execute one or more statements a specified number of times.

**Syntax:** ForNext variable, start, end, step, label
*statement-block*
label:

| Argument | Description |
|---|---|
| Variable | The name of the iteration counter variable having number data type. |
| Start | Starting value of variable. It can be a constant or name of another variable. |
| End | Ending value of variable. It can be a constant or name of another variable. |
| Step | The increment value. It can be a constant or name of another variable. |
| Label | An end loop label is an identifier followed by a colon (such as OK:). This label followed by a colon must be placed on a separate line. Do not use the colon with a label in the LoopWhile statement. |

**Usage:**

- Using the start and end parameters: for a positive increment (step), end must be greater than start; for a negative increment, end must be less than start, otherwise it will lead to an infinite loops. When increment is positive and start is greater than end, *statement-block* does not execute. When increment is negative and start is less than end, *statement-block* does not execute.
- Use Break, GoTo of If statement to jump out of the loop structure. Note that Exit or Reboot statements will also interrupt the loop.
- Use Break statement to interrupt the loop and continue with the first statement that follows the loop end label.
- Use GoTo label inside loop structure (where the label is the same as the label referenced in ForNext) to jump to the end of the loop and continue looping. You can also use Continue statement for this purpose.
- The 24x7 Scheduler does not perform iteration when program control is transferred into a loop structure from outside of it.
- 24x7 Scheduler supports nested loops. A nested loop is a loop placed inside another loop.
- The label name must satisfy Universal Naming Convention (UNC).

**Note:**
When start and end are variables, they are reevaluated on each pass through the loop. So if the variable's value changes, it will affect the number of loops. Consider this example—the body of the loop changes the number of rows in the database buffer, which changes the result of the DatabaseRowCount statement:

```
...
DatabaseRowCount( row_count )
ForNext( n, 1, row_count, 1, END_LOOP )
    DatabaseDelete( 1 )
    Subtract( row_count, 1, row_count )
END_LOOP:
...
```

**See also:**

LoopUntil
LoopWhile
Break

Continue

# GoTo

**Description:** Transfers control from this statement in a script to another statement or statement that is preceded by a label.

**Syntax:** GoTo label

| Argument | Description |
|---|---|
| Label | The label associated with the statement or statement to which you want to transfer control. A label is an identifier followed by a colon (such as OK:). This label followed by a colon must be placed on a separate line. Do not use the colon with a label in the GoTo statement. |

**Usage:** The label name must satisfy Universal Naming Convention (UNC).

**See also:**

IfThen
Exit
OnErrorGoTo
Break

# If

**Description:** Evaluates a conditional Boolean variable and depending on its value transfers control to the statement following the specified labels.

**Syntax:**
If ( boolean, truelabel, falselabel )
    statement-block
truelabel:
    statement-block
falselabel:
    statement-block

| Argument | Description |
|---|---|
| Boolean | The boolean variable that evaluates to TRUE or FALSE. If the boolean variable evaluates to TRUE then control will be transferred to the statement going after truelabel, otherwise control will be transferred to the statement going after falselabel. |
| statement-block | The block of JAL statements you want 24x7 Scheduler to execute. |
| Truelabel | A truelabel is an identifier followed by a colon (such as |

| | |
|---|---|
| | OK:). This truelabel followed by a colon must be placed on a separate line |
| Falselabel | A falselabel is an identifier followed by a colon (such as OK:). This falselabel followed by a colon must be placed on a separate line. |

**Usage:**
You can use If statements to branch program logic that depends on some conditions. The truelabel and falselabel names must satisfy Universal Naming Convention (UNC).

**Notes:**
- Instead of the boolean value you can specify numeric value. In case of numeric value, 0 (zero) always evaluates to FALSE, otherwise it evaluates to TRUE.
- After executing block of JAL statements following truelabel 24x7 Scheduler executes section following falselabel unless you code some other Control-Of-Flow statement to transfer control beyond falselabel section.

**See also:**

IfThen

# IfThen

**Description:** Evaluates a conditional Boolean variable and depending on its value transfers control to the next statement in script or the statement following the specified label.

**Syntax:**
IfThen ( boolean, label)
    statement-block
label:

| Argument | Description |
|---|---|
| boolean | The boolean variable that evaluates to TRUE or FALSE. If the boolean variable evaluates to FALSE then control will be transferred to the next statement going after IfThen, otherwise control will be transferred to the statement going after the label. |
| statement-block | The block of JAL statements you want 24x7 Scheduler to execute if the boolean value is False. |
| label | A label is an identifier followed by a colon (such as OK:). This label followed by a colon must be placed on a separate line. |

**Usage:**
You can use If statements to branch program logic that depends on some conditions. The label name must satisfy Universal Naming Convention (UNC).

**Note:**
Instead of the boolean value you can specify numeric value. In case of numeric value, 0 (zero) always evaluates to FALSE, otherwise it evaluates to TRUE.

**See also:**

If

# ChooseCase

**Description:** Evaluates a conditional variable and depending on its value transfers control to the appropriate "Case" section.

**Syntax:**
ChooseCase testvariable, label
Case valuelist1
    statement-block
[Case valuelist2]
    statement-block
[Case ...]
    statement-block
[CaseElse]
    statement-block
label:

| Argument | Description |
|---|---|
| testvariable | The variable whose value is evaluated in the following Case sections. |
| Case | The keyword beginning the new Case section. You can define as many Case sections as needed. |
| valuelist | One of the following:<br>• A single value<br>• A list of values separated by commas (such as 2, 4, 6, 8)·<br>• A single variable<br>• A list of variables separated by commas (such as VAR1, VAR2, VAR5)·<br>• Any combination of the above with an implied OR between items (such as 1, 3, VAR2, 7, 9, VAR5)<br>Data types of values and variables in the list must match data type of the conditional testvariable |
| statement-block | The block of JAL statements you want 24x7 Scheduler to execute if the value of the testvariable matches the value in valuelist of the corresponding Case section. |
| CaseElse | The keyword beginning the CaseElse section to which control is transferred when the value of the testvariable does not match any of the valuelist |
| label | A label is an identifier followed by a colon (such as OK:). This label followed by a colon must be placed on a separate line. |

**Usage:**
You can use ChooseCase statements to branch program logic that depends on some conditions. The label name must satisfy Universal Naming Convention (UNC).

At least one Case section is required. You must end the ChooseCase control structure with the label.
If the value of the testvariable at the beginning of the ChooseCase statement matches a value in valuelist for a Case section, the statements immediately following that Case section are executed. Control then passes to the first statement after the label.

If multiple Case section exist, then testvariable is compared to each valuelist until a match is found or the CaseElse section or label is encountered.

CaseElse section is optional. If there is a CaseElse section and the testvariable value does not match any of the valuelist, statemens in the CaseElse Section are executed. If no CaseElse section exists and a match is not found, the first statement after the label is executed.

**Note:**
Data type of the testvariable can be any of JAL standard data types. Data types of values and variables in the valuelist must match data type of the testvariable

**See also:**

If

IfThen

# LoopWhile

**Description:** A control structure that is a general-purpose iteration statement used to execute a block of statements while a conditional Boolean variable evaluates to TRUE.

**Syntax:**
LoopWhile boolean, label
    *statement-block*
label:

| Argument | Description |
|---|---|
| Boolean | The boolean variable that evaluates to TRUE or FALSE. If the boolean variable evaluates to TRUE then control will be transferred to the statement going after LoopWhile, otherwise control will be transferred to the statement or statement going after label. |
| Statement-block | The block of statements you want to repeat. |
| Label | An end loop label is an identifier followed by a colon (such as OK:). This label followed by a colon must be placed on a separate line. |

**Usage:**
- You can use LoopWhile statements to branch program logic that depends on some conditions.
- Avoid infinite loops. Make sure to change the conditional Boolean variable so it evaluates to FALSE or use Break, GoTo of If statement to jump out of the loop structure. Note that Exit or Reboot statements will also interrupt the loop.
- Use Break statement to interrupt an endless loop and continue with the first statement that follows the loop end label.
- Use GoTo label inside loop structure (where the label is the same as the label referenced in LoopWhile) to jump to the end of the loop and continue looping. You can also use Continue statement for this purpose.

- The 24x7 Scheduler does not perform iteration when program control is transferred into a loop structure from outside of it.
- 24x7 Scheduler supports nested loops. A nested loop is a loop placed inside another loop.
- The label name must satisfy Universal Naming Convention (UNC).

**Note:**
Instead of the boolean value you can specify numeric value. In case of numeric value, 0 (zero) always evaluates to FALSE, otherwise it evaluates to TRUE.

**See also:**

LoopUntil
ForNext
Break
Continue

# LoopUntil

**Description:** A control structure that is a general-purpose iteration statement used to execute a block of statements while a conditional Boolean variable does not evaluate to TRUE.

**Syntax:**
LoopUntil boolean, label
*statement-block*
label:

| Argument | Description |
|---|---|
| Boolean | The boolean variable that evaluates to TRUE or FALSE. If the boolean variable evaluates to FALSE then control will be transferred to the statement going after LoopUntil, otherwise control will be transferred to the statement or statement going after label. |
| Statement-block | The block of statements you want to repeat. |
| Label | An end loop label is an identifier followed by a colon (such as OK:). This label followed by a colon must be placed on a separate line. |

**Usage:**
- You can use LoopUntil statements to branch program logic that depends on some conditions.
- Avoid infinite loops. Make sure to change the conditional Boolean variable so it evaluates to TRUE or use Break, GoTo of If statement to jump out of the loop structure. Note that Exit or Reboot statements will also interrupt the loop.
- Use Break statement to interrupt an endless loop and continue with the first statement that follows the loop end label.
- Use GoTo label inside loop structure (where the label is the same as the label referenced in LoopUntil) to jump to the end of the loop and continue looping. You can also use Continue statement for this purpose.
- The 24x7 Scheduler does not perform iteration when program control is transferred into a loop structure from outside of it.
- 24x7 Scheduler supports nested loops. A nested loop is a loop placed inside another loop.

**Note:**
Instead of the boolean value you can specify numeric value. In case of numeric value, 0 (zero) always evaluates to FALSE, otherwise it evaluates to TRUE.

**See also:**

LoopWhile
ForNext
Break
Continue

# OnErrorGoTo

**Description:** Instructs 24x7 Scheduler in case of run-time script error to continue script execution with the statement immediately following the label specified in the OnErrorGoTo statement. This error handling behavior applies only to the portion of the script whose execution follows OnErrorGoTo statement.

**Syntax:** OnErrorGoTo label

| Argument | Description |
|---|---|
| label | The label associated with the statement or statement to which you want to transfer control in case if a run-time error occurs. A label is an identifier followed by a colon (such as OK:). This label followed by a colon must be placed on a separate line. Do not use the colon with a label in the OnErrorGoTo statement. |

**Usage:** The label name must satisfy Universal Naming Convention (UNC).

OnErrorGoTo statement overrides default error handling specified by the **Ignore Errors** job property. It also overrides error handling mode previously set using OnErrorStop and OnErrorResumeNext statements (if any of them were used).

**Tip:** Use OnErrorGoTo statements to implement custom job error handling procedures. For example:

```
OnErrorGoTo label_1
    statement-block
OnErrorGoTo label_2
    statement-block
OnErrorGoTo label_3
    statement-block
Exit

label_1:
    statement-block
    Exit
label_2:
    statement-block
    Exit
label_3:
    statement-block
    Exit
```

**See also:**

RaiseError
OnErrorResumeNext
OnErrorStop
Exit

# OnErrorResumeNext

**Description:** Instructs 24x7 Scheduler to ignore run-time script errors and continue script execution whenever a run-time error is detected. This error handling behavior applies only to the portion of the script whose execution follows OnErrorResumeNext statement. The code execution continues with the statement immediately following the statement that caused the run-time error.

**Syntax:** OnErrorResumeNext

**Usage:** This statement has no arguments. OnErrorResumeNext overrides default error handling specified by the **Ignore Errors** job property. It also overrides error handling mode previously set using OnErrorStop and OnErrorGoTo statements (if any of them were used).

**See also:**

RaiseError
OnErrorGoTo
OnErrorStop
Exit

# OnErrorStop

**Description:** Instructs 24x7 Scheduler to stop the script and generate an error message whenever a run-time error is detected. This error handling behavior applies only to the portion of the script whose execution follows OnErrorStop statement.

**Syntax:** OnErrorStop

**Usage**: This statement has no arguments. OnErrorStop overrides default error handling specified by the **Ignore Errors** job property. It also overrides error handling mode previously set using OnErrorResumeNext and OnErrorGoTo statements (if any of them were used).

If you do not use OnErrorStop and other OnError statements in your code, the error handling is completely controlled by the **Ignore Errors** job property.

**See also:**

RaiseError
OnErrorGoTo
OnErrorResumeNext
Exit

# GetLastError

**Description:** Returns calling job's last-error message.

**Syntax:** GetLastError message

| Argument | Description |
|---|---|
| message | A string variable that receives the returned value. |

**Usage**: The last-error is maintained on a per-job basis. Multiple jobs do not overwrite each other's last-error.

**See also:**

RaiseError
OnErrorGoTo
OnErrorResumeNext
OnErrorStop

# Set

**Description:** Assigns new value to a variable.

**Syntax:** Set variable, value

| Argument | Description |
|---|---|
| variable | The name of variable you want to change |
| value | The new value that will be assigned to the variable. The value data type must be the same as variable data type |

**Usage**: The following data types are supported: String, Number, Boolean, Date, Time, DateTime

# Bitwise statements

## BitwiseAnd

**Description:** Performs a bitwise AND operation on each bit of two passed values.

**Syntax:** BitwiseAnd value1, value2, return

| Argument | Description |
|---|---|
| value1 | A first number whose value is used in the bitwise AND operation |
| value2 | A second number whose value is used in the bitwise AND operation |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns the result of the AND operation on each of value1's and value2's bits.

**Usage:** In a bitwise AND operation:
1 AND 0 evaluates to 0
0 AND 0 evaluates to 0
0 AND 1 evaluates to 0
1 AND 1 evaluates to 1

**See also:**

Bitwise statements

## BitwiseClearBit

**Description:** Clears the specified bit in a number.

**Syntax:** BitwiseClearBit value, bit, return

| Argument | Description |
|---|---|
| value | A number whose value is used in the bitwise operation |
| bit | A number whose value is the bit number that you want to clear |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns the result of the bitwise operation.

**Usage:** If the specified bit is off (0), the returned value equals the original value.

**See also:**

Bitwise statements

## BitwiseFlipBit

**Description:** Reverses the specified bit in a number.

**Syntax:** BitwiseFlipBit value, bit, return

| Argument | Description |
|---|---|
| value | A number whose value is used in the bitwise operation |
| bit | A number whose value is the bit number that you want to flip |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns the result of the bitwise operation.

**Usage:** If the specified bit is off (0), BitwiseFlipBit sets the bit on (1), otherwise it clears the bit.

**See also:**

Bitwise statements

## BitwiseGetBit

**Description:** Reports whether the specified bit is on (1) or off (0).

**Syntax:** BitwiseGetBit value, bit, return

| Argument | Description |
|---|---|
| value | A number whose value is used in the bitwise operation |
| bit | A number whose value is the bit number that you want to test |
| return | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE if the bit is on (1) and FALSE if it is off (0).

**See also:**

Bitwise statements

# BitwiseOr

**Description:** Performs a bitwise OR operation on each bit of two passed values.

**Syntax:** BitwiseOr value1, value2, return

| Argument | Description |
|---|---|
| value1 | A first number whose value is used in the bitwise OR operation |
| value2 | A second number whose value is used in the bitwise OR operation |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns the result of the OR operation on each of value1's and value2's bits.

**Usage:** In a bitwise OR operation:
1 OR 0 evaluates to 1
0 OR 0 evaluates to 0
0 OR 1 evaluates to 1
1 OR 1 evaluates to 1

**See also:**

Bitwise statements

# BitwiseNot

**Description:** Performs a bitwise NOT operation on each bit of a passed value.

**Syntax:** BitwiseNot value, return

| Argument | Description |
|---|---|
| value | A number whose value is used in the bitwise NOT operation |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns the result of the bitwise NOT operation on each of value's bits.

**Usage:** In a bitwise NOT operation:
1 evaluates to 0
0 evaluates to 1

**See also:**

Bitwise statements

## BitwiseSetBit

**Description:** Sets the specified bit on (1) in a number.

**Syntax:** BitwiseSetBit value, bit, return

| Argument | Description |
|---|---|
| value | A number whose value is used in the bitwise operation |
| bit | A number whose value is the bit number that you want to set |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns the result of the bitwise operation.

**Usage:** If the specified bit is on (1), the returned value equals the original value.

**See also:**

Bitwise statements

## BitwiseXor

**Description:** Performs a bitwise XOR operation on each bit of two passed values.

**Syntax:** BitwiseXor value1, value2, return

| Argument | Description |
|---|---|
| value1 | A first number whose value is used in the bitwise XOR operation |
| value2 | A second number whose value is used in the bitwise XOR operation |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns the result of the XOR operation on each of value1's and value2's bits.

**Usage:** In a bitwise XOR operation:
1 XOR 0 evaluates to 1
0 XOR 0 evaluates to 0
0 XOR 1 evaluates to 1
1 XOR 1 evaluates to 0

**See also:**

Bitwise statements

# Clipboard statements

## ClipboardGet

**Description:** Obtains a copy of the text stored in the system clipboard.

**Syntax:** ClipboardGet return

| Argument | Description |
|---|---|
| return | A string variable that receives the Clipboard data |

**Return value:** String. Returns a copy of the text data stored in the system clipboard if it succeeds and the empty string ("") if an error occurs.

**See also:**

ClipboardSet

## ClipboardSet

**Description:** Replaces a contents of the system clipboard with the specified text.

**Syntax:** ClipboardSet string

| Argument | Description |
|---|---|
| string | The string you want to copy to the clipboard |

**Return value:** None.

**See also:**

ClipboardGet

# Database statements

## DatabaseConnect

**Description:** Connects to a database using the specified profile.

**Syntax:** DatabaseConnect profile

| Argument | Description |
|---|---|
| profile | A string whose value is the name of the profile defined in the 24x7 Preferences |

**Return value:** None.

**Usage:** DatabaseConnect statement obtains from the profile all the required connection information for the database to which you want to connect. This statement must be executed before other database actions can be processed using the same profile. Only one database connection may be opened at a time. All other database statements executed after DatabaseConnect are sent to the database specified in the profile.

**See also:**

DatabaseDisconnect

Database Profiles

## DatabaseConnectEx

**Description:** Connects to the database using the specified connection parameters.

**Syntax:** DatabaseConnectEx dbms, server_name, database_name, user_id, password, auto_commit

| Argument | Description |
|---|---|
| DBMS | A string whose value is the type of the database management system that you want to connect to.<br>This name must match any supported DBMS name as they are specified in the Database Profiles dialog (see Tools/Database Profiles menu) |
| server_name (Optional) | A string whose value is 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 (DBMS=ODBC) specify name of the desired ODBC profile as the server_name |
| database_name | A string whose value is name of the database. Specify |

| | |
|---|---|
| (Optional) | an empty string "" if the database name is not required for connection. |
| user_id | A string whose value is the name of the user to log on to the database |
| password | A string whose value is the password to use to log on to the database |
| auto_commit | A boolean whose value controls 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. |

**Return value:** None.

**Usage:** DatabaseConnectEx or DatabaseConnect statement must be executed before other database actions can be processed.

One job may have only one database connection open at a time. However, multiple jobs may have multiple database connections opened simultaneously.

All other database statements executed after DatabaseConnectEx are sent to the database connected in the same job.

**Important Note:** In some databases AutoCommit is required in order to execute DDL statements and create temporary tables. For example, this is required in MS SQL Server and Sybase SQL Server.

**See also:**

DatabaseDisconnect

Database Profiles
DatabaseConnect

# DatabaseCopy

**Description:** Copies the data from the primary database buffer to the system clipboard.

**Syntax:** DatabaseCopy return

| Argument | Description |
|---|---|
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns the number of rows copied to the clipboard.

**Usage:** DatabaseCopy uses tab characters to separate columns and carriage return (CR), linefeed (LF) pairs to separate rows. The copied data can be easily pasted into a spreadsheet program or into other Windows applications that supports data format described above..
The data you want to copy can be previously retrieved from the database using DatabaseRetrieve statement or pasted from the system clipboard using DatabasePaste statement.

**See also:**

DatabasePaste
Clipboard statements

# DatabaseDelete

**Description:** Deletes a row from the database buffer.

**Syntax:** DatabaseDelete row

| Argument | Description |
|---|---|
| row | A number identifying the row you want to delete |

**Return value:** None.

**Usage:** The row is not deleted from the database table until you call the DatabaseUpdate statement. After the DatabaseUpdate statement has successfully updated the database, then the storage associated with the row is cleared.

**See also:**

DatabaseRetrieve
DatabaseRowCount
DatabaseSetFilter
DatabaseUpdate

# DatabaseDescribe

**Description:** Describes result set definition in the database buffer.

**Syntax:** DatabaseDescribe return

| Argument | Description |
|---|---|
| return | A string variable that receives the returned value |

**Return value:** String. Returns current result set description.

**Usage:** This statement is provided for debugging purposes. The 24x7 Scheduler creates result set definition while executing DatabaseSetSQLSelect or DatabaseRetrieve statements. You can use MessageBox statement to display the returned result set definition.

**See also:**

MessageBox
DatabaseRetrieve
DatabaseSetSQLSelect

# DatabaseDisconnect

**Description:** Disconnects from a database.

**Syntax:** DatabaseDisconnect

**Return value:** None.

**Usage:** DatabaseDisconnect statement has no arguments. This statement disconnects the 24x7 Scheduler from the database to which you previously connected using DatabaseConnect statement.

**See also:**

DatabaseConnect
Database Profiles

# DatabaseExecute

**Description:** Execute a SQL statement that does not produce a result set.

**Syntax:** DatabaseExecute sql, return

| Argument | Description |
|---|---|
| sql | A string whose value is a valid SQL command that you want to send to the database. |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns the number of rows affected, if applicable.

**Usage:**
Refer to your database documentation for the correct SQL syntax. DatabaseExecute sends the specified SQL as is. Before you execute a SQL, be sure you have active database connection. You use the DatabaseConnect statement to establish a database connection.

**See also:**

DatabaseRetrieve
DatabaseUpdate

# DatabaseExport

**Description:** Exports data from the specified database table and in stores it in the specified file.

**Syntax:** DatabaseExport table, file, return

| Argument | Description |
|---|---|
| table | A string whose value is the name of database table that you want to export |
| file | A string whose value is the name of the file into which you want to save exported data |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns the number of exported rows.

**See also:**

DatabaseRetrieve
DatabaseImport
DatabaseSave

# DatabaseGet

**Description:** Obtains the current value of a cell in the primary database buffer.

**Syntax:** DatabaseGet row, column, return

| Argument | Description |
|---|---|
| row | A number whose value is the row number for the cell whose value you want to get |
| column | A number whose value is the column number for the cell whose value you want to get |
| return | A string variable that receives the returned value |

**Return value:** String. Returns the string representation of the data value at the row and column location. If the value is null, DatabaseGet returns the empty string ("").

**See also:**

DatabaseSet
DatabaseUpdate

# DatabaseImport

**Description:** Imports data from the specified file and inserts it into the specified database table.

**Syntax:** DatabaseImport table, file, return

| Argument | Description |
|---|---|
| table | A string whose value is the name of database table into which you want to import data |
| file | A string whose value is the name of the file that you want to import |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns the number of imported rows.

**See also:**

DatabaseRetrieve
DatabaseExport

# DatabaseInsert

**Description:** Inserts an empty row into the database buffer.

**Syntax:** DatabaseInsert return

| Argument | Description |
|---|---|
| return | A number identifying the inserted row. The row is always inserted at the end of the dataset. |

**Return value:** Number. Returns the number of the new row.

**Usage:** A newly inserted row is not marked as modified until data is entered in the row. You can use the DatabaseSet statement to enter data in the newly inserted row. The new modified row is not inserted into the database table until you call the DatabaseUpdate statement. After the DatabaseUpdate statement has successfully updated the database, the row state changes to unmodified.

**See also:**

DatabaseRetrieve
DatabaseRowCount
DatabaseSet
DatabaseUpdate

# DatabasePaste

**Description:** Pastes the data from the system clipboard to the primary database buffer.

**Syntax:** DatabasePaste return

| Argument | Description |
|---|---|
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns the number of rows pasted from the clipboard.

**Usage:** The clipboard data is expected in ASCII text format with tab characters separating columns and carriage return (CR), linefeed (LF) pairs separating rows. The result set definition must exist in the database buffer before pasting data. Number of columns and their data types defined in the database buffer must match data in the clipboard. You can use either DatabaseSetSQLSelect or DatabaseRetrieve statement to create result set definition in the database buffer. You may want to call DatabaseUpdate statement after pasting data to save new data in the database.

**See also:**

DatabaseCopy

Clipboard statements

# DatabasePipe

**Description:** Copies data from the source database to the destination database as specified by the SQL query and update method.

**Syntax:** DatabasePipe source, destination, query, table, method, return

| Argument | Description |
|---|---|
| source | A string whose value is the profile name of the source database |
| destination | A string whose value is the profile name of the destination database |
| query | A string whose value a valid SQL SELECT statement that is used to retrieve data from the source database |
| table | A string whose value is the name of database table in the destination database into which you want to copy data from the source database |
| method | A string constant whose value specifies how you want to update the destination table. Values are:<br>• "INSERT" — Execute DELETE then INSERT SQL statements for each copied row<br>• "UPDATE" — Execute UPDATE SQL statement for each copied row |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns the number of transferred rows.

**Usage:** Before the 24x7 Scheduler executes the pipeline, it connects to the source and destination databases. This does not affect active database connection if any. When the 24x7 Scheduler executes the pipeline, the piped data is committed every 100 rows. The data is also committed when the pipeline finishes. If a database error occurs, last not committed yet data portion is rolled back.

**Caution:**
Special considerations for the “UPDATE" method:

- For each row to be updated you should have at least one matching row in the destination table. Otherwise a database error will occur. The match is based on the primary key values.
- The primary key must be defined for the destination table. DatabasePipe uses primary key definition when constructing the UPDATE SQL statements.

Special considerations for the “INSERT" method:

- DatabasePipe generates and executes the DELETE SQL statement for each new row. Then it generates the INSERT SQL statements for each new row. This method allows avoiding duplicate records inserted in the destination table. However, if there is a foreign key of type “DELETE CASCADE" defined for the destination table, the database will enforce such referential integrity and will delete all referencing records from the child table(s).
- The primary key must be defined for the destination table. DatabasePipe uses primary key definition when constructing the DELETE SQL statements.

**See also:**

DatabaseRetrieve
DatabaseExport
DatabaseImport
DatabaseSave

# DatabaseRetrieve

**Description:** Retrieves data from the database using the specified SQL query.

**Syntax:** DatabaseRetrieve query, return

| Argument | Description |
|---|---|
| query | A string whose value a valid SQL SELECT statement that is used to retrieve data |
| return | A numeric variable that receives the number of rows retrieved |

**Return value:** Number. Returns the number of retrieved rows.

**Usage:** After rows are retrieved, the database buffer’s filter and sort options are applied. Therefore, any retrieved rows that don't meet the filter criteria are not included in the returned data and number of rows. The retrieved data remains in the database buffer until new retrieval. A new call to the DatabaseFilter and/or DatabaseSort statements will affect data in the primary database buffer.
Before you retrieve rows, be sure you have active database connection. You use the DatabaseConnect statement to establish a database connection.

**See also:**

DatabaseExport
DatabaseConnect

# DatabaseRowCount

**Description:** Obtains the number of rows that are currently available in the primary database buffer. If there is a filter set on the database buffer, the DatabaseRowCount statement checks only rows that have been left.

**Syntax:** DatabaseRowCount return

| Argument | Description |
|---|---|
| return | A numeric variable that receives the number of rows retrieved |

**Return value:** Number. Returns the number of rows that are currently available in the primary database buffer, 0 if no rows are currently available, and -1 if an error occurs.

**Usage:** The number of currently available rows equals the total number of rows retrieved plus any pasted rows minus any rows that have been filtered out.

**See also:**

DatabaseSetFilter

# DatabaseSave

**Description:** Saves the contents of the primary database buffer as an external file in the specified format.

**Syntax:** DatabaseSave file, format, return

| Argument | Description |
|---|---|
| file | A string whose value is the name of the file in which to save the contents. If you specify an empty string (""), the 24x7 Scheduler prompts for the file name |
| format | A string constant specifying the format in which to save the contents of the primary database buffer. Values are:<br>"DBF" — dBASE-III format<br>"DIF" — Data Interchange Format<br>"XLS" — Microsoft Excel 5 format<br>"HTML" or "HTM" — table in HTML format<br>"SQL" — SQL syntax with CREATE TABLE statement following by INSERT statements<br>"TXT" — Text format with tab-separated columns and return at the end of each row<br>"CSV" — Text format with tab-separated columns and return at the end of each row<br>"WKS" — Lotus 1-2-3 format |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns the number of rows saved in the specified file.

**Usage:** The number of rows currently available for saving equals the total number of rows retrieved plus any pasted rows minus any rows that have been filtered out. If you specify an empty string for the file name, the 24x7 Scheduler displays the Save Rows As dialog. Execution of the script will not continue until you either enter missing file name interactively or cancel the dialog.

**See also:**

DatabaseExport
DatabaseRetrieve
DatabaseSetFilter

# DatabaseSet

**Description:** Changes the value of a cell in the primary database buffer to the specified value.

**Syntax:** DatabaseSet row, column, value

| Argument | Description |
|---|---|
| row | A number whose value is the row number for the cell whose value you want to change |
| column | A number whose value is the column number for the cell whose value you want to change |
| value | A string whose value is the string representation of the new value to which you want to set the data at the row and column location |

**Return value:** None.

**Usage:** The value is not changed in the database table until you call the DatabaseUpdate statement. DatabaseSet statement automatically converts the specified value to the appropriate data type that matches data type of the database table that you want to update.

**See also:**

DatabaseGet
DatabaseUpdate

# DatabaseSetFilter

**Description:** Changes filter criteria for the database buffer. Rows that do not meet the filter criteria are moved to the filter buffer.

**Syntax:** DatabaseSetFilter filter, return

| Argument | Description |
|---|---|
| filter | A string whose value is a boolean expression that you want to use as the filter criteria. The expression includes column names or numbers. A column number must be preceded by a pound sign (#). |

**Return value:** None.

**Usage:** The database buffer can have filter criteria specified as part of its definition. After data is retrieved, rows that don't meet the criteria are immediately transferred from the primary database buffer to the filter buffer.
The DatabaseSetFilter statement replaces the filter criteria defined for the database buffer, if any, with a new set of criteria. The new filter is applied immediately.

The filter expression consists of columns, functions, operators, and values against which column values are compared. Boolean expressions can be connected with AND and OR logical operators. You can also use NOT, the negation operator. Use parentheses to control the order of evaluation.

Sample expressions are:

```
item_id > 5
NOT item_id = 5
(NOT item_id = 5) AND customer > "Mabson"
item_id > 5 AND customer = "Smith"
left(customer, 5) = "Smith"
#1 > 5 AND #2 = "Smith"
```

The filter expression is a string and does not contain variables. However, you can build the string at during execution using the values of variables in the script. Within the filter string, values that are strings must be enclosed in quotation marks (see the examples).
If the filter expression contains numbers, the database buffer expects the numbers in U.S. format. Be aware that when converting numbers to string the String statement formats numbers using the current system settings. If you use it to build the filter expression, use the Format statement to specify the desired format.

To remove a filter, call DatabaseSetFilter with the empty string (""). The rows in the filter buffer will be restored to the primary database buffer after the rows that were already in the primary buffer.

**See also:**

DatabaseSetSort
Operators in Filter and Sort Expressions
Functions in Filter and Sort Expressions by Category

# DatabaseSetSort

**Description:** Changes sort criteria for the database buffer.

**Syntax:** DatabaseSetSort sort, return

| Argument | Description |
|---|---|
| sort | A string whose value is an expression that you want to use as the sort criteria. The expression includes column names or numbers. A column number must be preceded by a pound sign (#). |

**Return value:** None.

**Usage:** The database buffer can have sort criteria specified as part of its definition. After data is retrieved, the 24x7 Scheduler performs immediate sorting according to the current sort criteria.
The DatabaseSetSort statement replaces the sort criteria defined for the database buffer, if any, with a new set of criteria. The new sort is applied immediately.

The sort expression may include columns and functions.

The sort criteria for a column has one of the forms shown in the following table, depending on whether you specify the column by name or number. Order is either A for ascending or D for descending order. You can specify secondary sorting by specifying criteria for additional columns in the format string. Separate each column specification with a comma.

| Syntax for sort order | Examples |
|---|---|
| columnname order | "emp_lname A"<br>"emp_lname A, dept_id D" |
| # columnnumber order | "#3 A" |

In place of a column in the sort criteria you can use an expressions.

Sample expressions are:

"(item_id * 5) A"
if(status = 'A', current_rate, prev_rate * 01) A
left(#1, 5) = D, #3 A

The sort expression is a string and does not contain variables. However, you can build the string at during execution using the values of variables in the script. Within the sort string, values that are strings must be enclosed in quotation marks (see the examples).
If the sort expression contains numbers, the database buffer expects the numbers in U.S. format. Be aware that when converting numbers to string the String statement formats numbers using the current system settings. If you use it to build the filter expression, use the Format statement to specify the desired format.

To remove the database buffer sorting, call DatabaseSetSort with the empty string (""). However the empty string ("") does not restore original sort order made in the database at during retrieval of data.

**See also:**

DatabaseSetFilter
Operators in Filter and Sort Expressions
Functions in Filter and Sort Expressions by Category

# DatabaseSetSQLSelect

**Description:** Specifies the new SQL SELECT statement for the database buffer. The result set definition in the database buffer is updated to match the new SQL.

**Syntax:** DatabaseSetSQLSelect sql

| Argument | Description |
|---|---|
| sql | A string whose value is the new SQL SELECT statement that you want to use for the database operations |

**Return value:** None.

**Usage:** The DatabaseSetSQLSelect statement destroys data in the database buffer, if any, then creates the new result set definition based on supplied SELECT statement. Any existing filter and sort criteria are also reset.

**See also:**

DatabaseRetrieve

# DatabaseUpdate

**Description:** Updates the database with the changes made.

**Syntax:** DatabaseUpdate method, return

| Argument | Description |
|---|---|
| method | A string constant whose value specifies how you want to update the destination table. Values are:<br>• "INSERT" — Execute DELETE then INSERT SQL statements for each modified row<br>• "UPDATE" — Execute UPDATE SQL statement for each modified row |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns the number of newly updated/inserted/deleted rows.

**Usage:** The DatabaseUpdate statement generates UPDATE, INSERT, and DELETE SQL statements for all new or changed rows in the database buffer including rows in the filter buffer, if any. The DatabaseUpdate always generates the INSERT SQL statement for all new rows. Depends on the specified update method, DatabaseUpdate determines for each modified row, whether the row can be updated in place or whether the row has to be deleted and reinserted. The database changes are committed after the last row is updated. If a database error occurs, all database changes are rolled back.

**Caution:**
Special considerations for the “UPDATE" method:

- When there are multiple rows modified in the database buffer and you have switched keys or rows, updating in place may fail due to DBMS duplicate restrictions.
- The primary key must be defined for the destination table. DatabaseUpdate uses primary key definition when constructing UPDATE SQL statements.

Special considerations for the “INSERT" method:

- DatabaseUpdate generates and executes the DELETE SQL statement for each row that needs to be updated. Then it generates the INSERT SQL statements that reinserts each modified row in the database. This method allows avoiding duplicate records inserted in the destination table. However, if there is a foreign key of type “DELETE CASCADE" defined for the destination table, the database will enforce such referential integrity and will delete all referencing records from the child table(s).
- The primary key must be defined for the destination table. DatabaseUpdate uses primary key definition when constructing DELETE SQL statements.

**See also:**

DatabaseRetrieve

DatabaseImport

DatabasePipe

DatabasePaste
DatabaseDelete

# Date and time statements

## AtomicTime

**Description:** Obtains time from the specified atomic time server and optionally sets the local system time to the obtained value.

**Syntax:** AtomicTime ( server, set_local, result )

| Argument | Description |
|---|---|
| server | A string containing a server name or IP address. |
| set_local | A boolean value that controls whether AtomicTime statement should automatically update system time on the computer where it is running |
| Result | A datetime variable that receives the returned value |

**Return value:** DateTime. Returns atomic clock date and time.

**Usage:** AtomicTime statement utilizes standard time protocol called SNTP with UDP connections (RFC-1305). In theory it returns the exact time value which is accurate to 10ms or so. On practice the network traffic latency can slightly delay and thus affect the returned value.

There are many free atomic time servers connected to the Internet. You can use any server you like. For example you can use the following servers:

tick.usno.navy.mil
tmc.edu
time.nist.gov
utcnist.colorado.edu
time-nw.nist.gov
nist1.aol-ca.truetime.com
nist1.nyc.certifiedtime.com
nist1.sjc.certifiedtime.com
time-a.nist.gov
time-b.nist.gov
time-a.timefreq.bldrdoc.gov
time-b.timefreq.bldrdoc.gov
time-c.timefreq.bldrdoc.gov

You can use the AtomicTime statement to synchronize time on computer systems with precise time on atomic clock servers. You can also use the returned value to synchronize simultaneous jobs running on multiple computers.

**Note:**
In order to update the local system clock on Windows NT/2000/XP/2003/VISTA/2008/7 computers the 24x7 Scheduler must run under an administrator account.

**See also:**

HostTime statement
Today statement
Now statement
Timer statement

# Date

**Description:** Converts a string whose value is a valid date to a value of data type date.

**Syntax:** Date ( string, result )

| Argument | Description |
|---|---|
| String | A string containing a valid date (such as Jan 1, 1998, or 12-31-99) that you want returned as a date |
| Result | A date variable that receives the returned value |

**Return value:** Date. Returns the date in string as a date. If string does not contain a valid date, Date returns 1900-01-01.

**Usage:** The value of the string must be a valid date.

**Note:**
Valid dates in strings can include any combination of day (1 to 31), month (1 to 12 or the name or abbreviation of a month), and year (2 or 4 digits). 24x7 Scheduler assumes a 4-digit number is a year. Leading zeros are optional for month and day. The month, whether a name, an abbreviation, or a number, must be in the month location specified in the system setting for a date's format. If you do not know the system setting, use the standard data type date format "yyyy-mm-dd" .
A four-digit number is assumed to be a year. If the year is two digits, then 24x7 Scheduler chooses the century, as follows: If the year is between 00 and 49, program assumes 20 as the first two digits; if it is between 50 and 99, the 24x7 Scheduler assumes 19. If your data includes dates before 1950, such as birth dates, always specify a four-digit year so that the 24x7 Scheduler interprets the date as intended.

The 24x7 Scheduler handles years from 1000 to 3000 inclusive.

**See also:**

DateTime statement
MakeDate statement

# DateTime

**Description:** Combines a date and a time value into a DateTime value.

**Syntax:** DateTime ( date, time, result)

| Argument | Description |
|---|---|
| date | A valid date (such as Jan 1, 1998, or 12-31-99) that you want included in the value returned by DateTime |
| time | A valid time (such as 8AM or 10:25:23) that you want included in the value returned by DateTime. If you include a time, only the hour portion is required. If you omit the minutes, or seconds, they are assumed to be 0s. If you omit AM or PM, the hour is determined according to the 24-hour clock |
| result | A datetime variable that receives the returned value |

**Return value:** DateTime. Returns a DateTime value based on the values in date and time.

**Usage:** For information on valid dates, see Date statement.

## DateAdd

**Description:** Obtains the date that occurs a specified number of days after or before another date.

**Syntax:** DateAdd ( date, n, result )

| Argument | Description |
|---|---|
| date | A date value |
| n | An number indicating the number of days |
| result | A date variable that receives the returned value |

**Return value:** Date. Returns the date that occurs n days after date if n is greater than 0. Returns the date that occurs n days before date if n is less than 0.

**See also:**

TimeAdd

DateTimeAdd

## DateDiff

**Description:** Determines the number of days between two specified dates..

**Syntax:** DateDiff ( date1, date2, result )

| Argument | Description |
|---|---|
| date1 | A date value that is the start date of the interval being |

| | |
|---|---|
| | measured |
| date2 | A date value that is the end date of the interval |
| result | A numeric variable that receives the returned value |

**Return value:** Number. Returns the number of days date2 occurs after date1. If date2 occurs before date1, DaysDiff returns a negative number.

**See also:**

DateTimeDiff
TimeDiff

# DateTimeAdd

**Description:** Obtains the time that occurs a specified number of seconds after or before another time.

**Syntax:** DateTimeAdd ( datetime, n, result )

| Argument | Description |
|---|---|
| datetime | A datetime value |
| n | A long number of seconds |
| result | A datetime variable that receives the returned value |

**Return value:** DateTime. Returns the time that occurs n seconds after datetime if n is greater than 0. Returns the time that occurs n seconds before datetime if n is less than 0.

**See also:**

TimeAdd
DateAdd

# DateTimeDiff

**Description:** Determines the number of seconds one time occurs after another.

**Syntax:** DateTimeDiff ( datetime1, datetime2, result )

| Argument | Description |
|---|---|
| datetime1 | A datetime value that is the start time of the interval being measured |
| datetime2 | A datetime value that is the end time of the interval |

| | |
|---|---|
| result | A numeric variable that receives the returned value |

**Return value:** Number. Returns the number of seconds datetime2 occurs after datetime1. If datetime2 occurs before datetime1, TimeDiff returns a negative number.

**See also:**

DateDiff

TimeDiff

# DateTimePart

**Description:** Extracts the number containing the specified part of a given datetime, date, or string value.

**Syntax:** DateTimePart ( datetime, part, result )

| Argument | Description |
|---|---|
| datetime | A datetime (optionally date or string) value from which you want to extract specified interval of time |
| part | A string value that specifies the interval of time you want to return |
| result | A numeric variable that receives the returned value |

**Return value:** Number. Returns the interval of time extracted from datetime value.

**Settings:** The interval argument has these settings:

| Setting | Description |
|---|---|
| dd | day of month |
| dw | day of week |
| mm | Month |
| yy | Year |
| hh | Hour |
| mi | Minute |
| ss | Second |
| qq | Quarter |

# DayName

**Description:** Determines the day of the week in a date value and returns the weekday's name.

**Syntax:** DayName (date, result )

| Argument | Description |
|---|---|
| date | The date for which you want the name of the day |
| result | A string variable that receives the returned value |

**Return value:** String. Returns a string whose value is the name of the weekday (Sunday, Monday, and so on) for date.

**See also:**
DayName
Macro-parameters

## DayNumber

**Description:** Determines the day of the week of a date value and returns the number of the weekday.

**Syntax:** DayNumber (date, result )

| Argument | Description |
|---|---|
| date | The date from which you want the number of the day of the week |
| result | A numeric variable that receives the returned value |

**Return value:** Number. Returns an integer (1-7) representing the day of the week of date. Sunday is day 1, Monday is day 2, and so on.

**See also:**
DayNumber
Macro-parameters

## HostTime

**Description:** Obtains time from the specified network computer.

**Syntax:** HostTime ( host, result )

| Argument | Description |
|---|---|
| host | A string containing a computer name or IP address. |
| Result | A datetime variable that receives the returned value |

**Return value:** DateTime. Returns time on the remote computer. The result is returned as it is reported by the remote computer without adjusting to the local time zone.

**Usage:** HostTime statement utilizes standard TCP time service on port 13. Frequently this service is called as Daytime Protocol (RFC-867). The remote computer must listen on port 13, and respond to requests in either TCP/IP or UDP/IP formats. The standard does not specify an exact format for the Daytime Protocol, but requires that the time is sent using standard ASCII characters. Different computer systems use different time formats for returned values. The HostTime is capable to recognize and properly parse 2 most commonly used formats:

- JJJJJ YY-MM-DD HH:MM:SS TT
- DDD MMM DD HH:MM:SS YYYY

You can use the HostTime statement to synchronize simultaneous processing on multiple computers.
You can also use it to find out local times on remote computers located in different time zones.

Before you use the HostTime statement make sure that the remote computer listens on port 13 and also that it reports time in one of the 2 formats described above. Most UNIX servers by default provide support Daytime Protocol. On Windows 2000 and XP the Daytime service by default is not enabled. You can use the Services applet in the Windows Control Panel to enable and start this service.

**See also:**

AtomicTime statement
Today statement
Now statement
Timer statement

# MakeDate

**Description:** Combines numbers representing the year, month, and day into a date value.

**Syntax:** MakeDate ( year, month, day, result )

| Argument | Description |
|---|---|
| year | The 4-digit year (1000 to 3000) of the date |
| month | The 1- or 2-digit number for the month (1 to 12) of the year |
| day | The 1- or 2-digit number for the day (1 to 31) of the month |
| result | A date variable that receives the returned value |

**Return value:** Date. Returns the date specified by the numbers for year, month, and day as a date data type. If any value is invalid (out of the range of values for dates), MakeDate returns 1900-01-01.

# MakeDateTime

**Description:** Combines numbers representing the year, month, day, hour, minute, and second into a datetime value.

**Syntax:** MakeDateTime ( year, month, day, hour, minute, second, result )

| Argument | Description |
|---|---|
| year | The 4-digit year (1000 to 3000) of the date |
| month | The 1- or 2-digit number for the month (1 to 12) of the year |
| day | The 1- or 2-digit number for the day (1 to 31) of the month |
| hour | The number for the hour (00 to 23) of the time |
| minute | The number for the minutes (00 to 59) of the time |
| second | The number for the seconds (0 to 59) of the time |
| result | A datetime variable that receives the returned value |

**Return value:** DateTime. Returns the datetime value based on the values in year, month, day, hour, minute, and second arguments.

# MakeTime

**Description:** Combines numbers representing hours, minutes, and seconds into a time value.

**Syntax:** MakeTime ( hour, minute, second, result )

| Argument | Description |
|---|---|
| hour | The number for the hour (00 to 23) of the time |
| minute | The number for the minutes (00 to 59) of the time |
| second | The number for the seconds (0 to 59) of the time |
| result | A time variable that receives the returned value |

**Return value:** Time. Returns the time as a time data type and 00:00:00 if the value in any argument is not valid (out of the specified range of values).

# Now

**Description:** Obtains the system time

**Syntax:** Now ( result )

| Argument | Description |
|---|---|
| result | A time variable that receives the returned value |

**Return value:** Time. Returns the current system time.

**See also:**
Today
Timer
Macro-parameters

# Timer

**Description:** Reports the amount of CPU time that has elapsed since specified starttime.

**Syntax:** Timer starttime, result

| Argument | Description |
|---|---|
| starttime | A number whose value is start of the time interval expressed in milliseconds |
| result | A number variable that receives the returned value. |

**Return value:** Number. Returns number of milliseconds elapsed since starttime. If you specify 0 for starttime the Timer statement returns number of milliseconds elapsed since 24x7 Scheduler startup.

**See also:**
Today
Now
Macro-parameters

# Time

**Description:** Converts a string to a time data type.

**Syntax:** Time ( string, result )

| Argument | Description |
|---|---|
| string | A string containing a valid time (such as 8AM or 10:25) that you want returned as a time data type. Only the hour is required; you do not have to include the minutes, or seconds of the time or AM or PM. The default value for minutes and seconds is 00 . AM or PM is determined automatically |
| result | A time variable that receives the returned value |

**Return value:** Time. Returns the time in string as a time data type. If string does not contain a valid time, Time returns 00:00:00.

## TimeAdd

**Description:** Obtains the time that occurs a specified number of seconds after or before another time.

**Syntax:** TimeAdd ( time, n, result )

| Argument | Description |
|---|---|
| time | A time value |
| n | A long number of seconds |
| result | A time variable that receives the returned value |

**Return value:** Time. Returns the time that occurs n seconds after time if n is greater than 0. Returns the time that occurs n seconds before time if n is less than 0. The maximum return value is 23:59:59.

**See also:**

DateAdd
DateTimeAdd

## TimeDiff

**Description:** Determines the number of seconds one time occurs after another.

**Syntax:** TimeDiff ( time1, time2, result )

| Argument | Description |
|---|---|
| time1 | A time value that is the start time of the interval being measured |

| | |
|---|---|
| time2 | A time value that is the end time of the interval |
| result | A numeric variable that receives the returned value |

**Return value:** Number. Returns the number of seconds time2 occurs after time1. If time2 occurs before time1, TimeDiff returns a negative number.

**See also:**

DateTimeDiff

DateDiff

## Today

**Description:** Obtains the system date.

**Syntax:** Today ( result )

| Argument | Description |
|---|---|
| result | A date variable that receives the returned value |

**Return value:** Date. Returns the current system date.

**See also:**

Now

Macro-parameters

# DDE statements

## DDEClose

**Description:** Closes a channel to a DDE server application that was opened by calling the DDEOpen statement.

**Syntax:** DDEClose channel

| Argument | Description |
|---|---|
| channel | A number whose value is the channel number previously returned by the DDEOpen statement that started the DDE conversation |

**Return value:** None.

**Usage:** Use DDEClose to close a channel to a DDE server application that was opened by calling the DDEOpen statement .

**Important notes:** DDE statements are not supported in **asynchronous** jobs and **remote** jobs running on the 24x7 Remote Agents. DDE statements are supported in **synchronous** jobs and **asynchronous detached** jobs.

**See also:**

DDEOpen

# DDEExecute

**Description:** Asks a DDE server application to execute the specified command.

**Syntax:** DDEExecute channel, command

| Argument | Description |
|---|---|
| channel | A number whose value is the channel number previously returned by the DDEOpen statement that started the DDE conversation |
| command | A string whose value is the command you want a DDE server application to execute. The format of the command depends on the DDE application you want to execute the command |

**Return value:** None.

**Usage:** The DDE server application must already be running when you call a DDE statement. Use the Run statement to start the application if necessary. Before using this statement, call DDEOpen to establish a DDE channel.

**Important notes:** DDE statements are not supported in **asynchronous** jobs and **remote** jobs running on the 24x7 Remote Agents. DDE statements are supported in **synchronous** jobs and **asynchronous detached** jobs.

**See also:**

DDEOpen

# DDEGetData

**Description:** Asks a DDE server application to provide data and stores that data in the specified variable.

**Syntax:** DDEGetData channel, location, return

| Argument | Description |
|---|---|
| channel | A number whose value is the channel number previously returned by the DDEOpen statement that started the DDE conversation |
| location | A string whose value is the location of the data you want returned. The format of the location depends on the DDE application that will receive the request |
| return | A string variable that receives the returned value |

**Return value:** String.

**Usage:** The DDE server application must already be running when you call a DDE statement. Use the Run statement to start the application if necessary. Before using this statement, call DDEOpen to establish a DDE channel.

**Important notes:** DDE statements are not supported in **asynchronous** jobs and **remote** jobs running on the 24x7 Remote Agents. DDE statements are supported in **synchronous** jobs and **asynchronous detached** jobs.

**See also:**

DDEOpen

# DDEOpen

**Description:** Opens a channel to a DDE server application.

**Syntax:** DDEOpen application, topic, return

| Argument | Description |
|---|---|
| application | A string specifying the DDE name of the DDE server application |
| topic | A string identifying the data or the instance of the application you want to use (for example, in Microsoft Excel, the topic name could be System or the name of an open spreadsheet) |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns the handle to the channel if it succeeds. If an error occurs, DDEOpen returns a negative number.

**Usage:** The DDE server application must already be running when you call a DDE statement. Use the Run statement to start the application if necessary.

**Important notes:** DDE statements are not supported in **asynchronous** jobs and **remote** jobs running on the 24x7 Remote Agents. DDE statements are supported in **synchronous** jobs and **asynchronous detached** jobs.

**See also:**

DDEClose

## DDESetData

**Description:** Asks a DDE server application to accept data and store it in the specified location.

**Syntax:** DDESetData channel, location, data

| Argument | Description |
|---|---|
| channel | A number whose value is the channel number previously returned by the DDEOpen statement that started the DDE conversation |
| location | A string whose value is the location of the data you want returned. The format of the location depends on the DDE application that will receive the request |
| data | A string whose value you want to send to the DDE server application |

**Return value:** None.

**Usage:** The DDE server application must already be running when you call a DDE statement. Use the Run statement to start the application if necessary. Before using this statement, call DDEOpen to establish a DDE channel.

**Important notes:** DDE statements are not supported in **asynchronous** jobs and **remote** jobs running on the 24x7 Remote Agents. DDE statements are supported in **synchronous** jobs and **asynchronous detached** jobs.

**See also:**

DDEOpen

# Email, Page, and Network statements

## MailConfig

**Description:** Overrides global email settings and setting various parameters for subsequent Mail operations in the current job.

**Syntax:** MailConfig property, new_value

| Argument | Description |
|---|---|
| Property | A string whose value is the name of the property for the FTP session that you want to change. The following properties are supported: |

| | |
|---|---|
| | • "MAIL INTERFACE"<br>• "PORT"<br>• "SERVER"<br>• "ENCODING"<br>• "FORMAT" |
| New_value | A string whose value is the new value for the property that you want to change.<br><br>The following values are supported for the "MAIL INTERFACE" property:<br>• "MAPI"<br>• "SMTP"<br>• "Lotus Notes"<br>The default value is "MAPI". The specified "MAIL INTERFACE" is used for all subsequent Mail commands executed in the same job.<br>The "PORT" parameter is applicable tto SMTP interface only and ignore for all other mail interfaces. Use the "PORT" parameter, to specify which SMTP port your want to use for all subsequent Mail commands executed in the same job. This setting applies to all statements that begin with Mail prefix: **MailSend** and **MailSendWithAttachment** statements. If you do not change this property, the default SMTP port is used for Mail operations.<br><br>The following values are supported for the "ENCODING" property:<br>• "none"<br>• "mime"<br>• "base64"<br>• "uuencode"<br>The default value is "none". Use this property, to specify SMTP message encoding format for all subsequent Mail commands executed in the same job. This property is used only when using SMTP mail interface.<br><br>The following values could be specified for the "FORMAT" property:<br>• "text/plain"<br>• "text/html"<br>• "text/xml"<br>• Any other valid message format that the email recipient's program can recognize<br>The default value is "text/plain". Use this property, to specify SMTP message format for all subsequent Mail commands executed in the same job. This property is used only when using SMTP mail interface.<br>command and this feature must not be disabled. |

**Return value:** None.

**Usage:** This statement should be primarily used in standalone command line JAL scripts. JAL commands executed within script type jobs should use global parameters set for the 24x7 Scheduler.

**See also:**
MailSend
MailSendWithAttachment

# MailSend

**Description:** Establishes a new mail session and sends the specified mail message. If the message information is incomplete, the mail system displays a dialog box that you use to enter missing information.

**Syntax:** MailSend profile, password, recipient, subject, message

| Argument | Description |
|---|---|
| profile | The profile value depends on the email system interface that you have selected in the 24x7 Scheduler Options.<br><br>MAPI: a string whose value is the profile name to use when starting a new MAPI session<br><br>Lotus Notes: a string whose value is the user name to use when logging to Lotus Notes<br><br>SMTP: a string whose value is the valid sender's email address |
| password | A string whose value is the user's mail system password |
| recipient | A string variable whose value is the name of the recipient for the message. |
| subject | A string variable whose value is the subject line, displayed in the message header |
| message | A string variable whose value is the content of the message body |

**Return value:** None.

**Usage:** If one or more message parameters are missing or incorrect, the mail system displays a dialog box. Execution of the script will not continue until you either enter missing information interactively or cancel the dialog. You can specify multiple recipients in one message. Use comma (,) to separate recipient names.

**See also:**
MailSendWithAttachment
PageSend
NetworkSend

# MailSendWithAttachment

**Description:** Establishes a new mail session and sends a mail message. If no message information is supplied, the mail system provides a dialog box for entering it before sending the message. An additional file can attached to the message.

**Syntax:** MailSendWithAttachment profile, password, recipient, subject, message, file

| Argument | Description |
|---|---|
| profile | The profile value depends on the email system interface that you have selected in the 24x7 Scheduler Options.<br><br>MAPI: a string whose value is the profile name to use when starting a new MAPI session<br><br>Lotus Notes: a string whose value is the user name to use when logging to Lotus Notes<br><br>SMTP: a string whose value is the valid sender's email address |
| password | A string whose value is the user's mail system password |
| recipient | A string variable whose value is the name of the recipient for the message. |
| subject | A string variable whose value is the subject line, displayed in the message header |
| message | A string variable whose value is the content of the message body |
| file | A string variable whose value is the full file name of the file you want to attach |

**Return value:** None.

**Usage:** If one or more message parameters are missing or incorrect, the mail system displays a dialog box. Execution of the script will not continue until you either enter missing information interactively or cancel the dialog.
You can specify multiple recipients in one message. Use comma (,) to separate recipient names.
You can also specify multiple attachments in one message. Use comma (,) to separate attachment file names.

**See also:**

MailSend

# PageSend

**Description:** Sends pager message using SNPP protocol (RFC 1861).

**Syntax:** PageSend pager, message

| Argument | Description |
|---|---|
| pager | A string variable whose value is the recipient's pager number in the format required by your carrier. |
| message | A string variable whose value is the message to be sent |

**Return value:** None.

**Usage:** PageSend transmits the message using SNPP server specified in **System Options**
You can specify multiple page recipients in one message. Use comma (,) to separate pager numbers.

The maximum size of the message is usually limited to about 100 characters. Contact your carrier to find out the exact limit.

**See also:**

MailSend
NetworkSend

# NetworkSend

**Description:** Sends network message to a network user or computer.

**Syntax:** NetworkSend recipient, message

| Argument | Description |
|---|---|
| recipient | A string variable whose value is the name of a network user or network computer in domain\user or domain\computer format, for example, NSNTSVR\Administrator |
| message | A string variable whose value is the message to be sent |

**Return value:** None.

**Usage:** You can specify multiple page recipients in one message. Use comma (,) to separate recipient names.

NetworkSend transmits the message just like the Windows NT  NET SEND command. However it does not support real message broadcasting. Each message recipient must be specified separately or as part of the recipient list.

In order to receive the message each recipient must be running the Windows NT Messenger service, which is usually installed and enabled by default. No error occurs if the recipient of the message is specified correctly but the messenger service or destination computer is not running.

The message is displayed on the recipient's computer in a modal system message box.

**Important Note:**
NetworkSend is not supported on Windows 95/98/Me systems

**See also:**

MailSend
PageSend

# File statements

## CD

**Description:** Changes the current directory for the current process.

**Syntax:** CD dir

| Argument | Description |
|---|---|
| dir | A string that is the directory you want to set as current directory. |

**Usage:** The dir specifies the path to the new current directory. This parameter may be a relative path or a fully qualified path. In either case, the fully qualified path of the specified directory is calculated and stored as the current directory. This new setting will affect all file operations that do not specify full paths to the referenced files.

**Notes:**
Each process has a single current directory made up of two parts:

- A disk designator that is either a drive letter followed by a colon, or a server name and share name (\\servername\sharename)
- A directory on the disk designator
- **Changing current directory may affect other already running jobs.**

## Dir

**Description:** Returns comma-separated list of files in the specified directory.

**Syntax:** Dir file_mask, return

| Argument | Description |
|---|---|
| File_mask | A string whose value is the DOS file mask that you want to use to list file. File_mask can contain wildcard characters (* and ?). File_mask can contain full or partial file path. |
| Return | A string variable that receives the returned value. |

**Return value:** String. Returns comma-separated list of files.

**Usage:** The Dir statement is equivalent to DOS *dir* command.
To get the list of all files use *.* file mask. If you don't include file path to the file_mask then Dir statement returns files from the current working directory.

**See also:**
- DirEx
- CD
- FileSearchEx
- FileFindFirst
- FTPDir
- RemoteDir

# DirEx

**Description:** Returns comma-separated list of files in the specified directory.

**Syntax:** Dir file_mask, return

| Argument | Description |
|---|---|
| File_mask | A string whose value is the DOS file mask that you want to use to list file. File_mask can contain wildcard characters (* and ?). File_mask can contain full or partial file path. |
| Return | A string variable that receives the returned value. |

**Return value:** String. Returns comma-separated list of full file names.

**Usage:** The DirEx statement is similar to the Dir statement. The only difference is that it returns list of full file names including file path while the Dir statement returns list of file names without path. For more information see the topic for the Dir statement.
To get the list of all files use *.* file mask. If you don't include file path to the file_mask then DirEx statement returns files from the current working directory.

**See also:**
- Dir
- CD
- FileSearchEx
- FileFindFirst
- FTPDir
- RemoteDir

# SubDir

**Description:** Returns comma-separated list of subdirectories in the specified directory.

**Syntax:** SubDir path, return

| Argument | Description |
|---|---|
| path | A string whose value is the file path where you want to search for subdirectories. |
| Return | A string variable that receives the returned value. |

**Return value:** String. Returns comma-separated list of subdirectory names.

**Usage:** The SubDir statement is similar to the Dir statement. The only difference is that it returns list of subdirectory names while the Dir statement returns complete list of all file and subdirectory names that match the specified criteria.

**See also:**

Dir
CD
FileSearchEx
FileFindFirst
FTPDir
RemoteDir

# DirCreate

**Description:** Creates a new directory.

**Syntax:** DirCreate path

| Argument | Description |
|---|---|
| path | A string whose value is either absolute or relative path of the directory to be created |

**Return value:** None

**Usage:** DirCreate statement can recursively create all missing subdirectories referenced in the path. For example, if DirCreate is called with *C:\subdir1\subdir2\subdi3* parameter and subdirectories *subdir2* and *subdir3* do not exist DirCreate will create both of them.

**See also:**

DirDelete
isDir
CD

# DirDelete

**Description:** Deletes an existing directory and all files in it. If that directory contains subdirectories, 24x7 Scheduler deletes subdirectories recursively.

**Syntax:** DirDelete dir

| Argument | Description |
|---|---|
| dir | A string whose value is the name of the directory you want delete |

**Return value:** None

**See also:**

isDir
FileDelete
FileDeleteEx
DirCreate

# DirEx

**Description:** Returns comma-separated list of files in the specified directory.

**Syntax:** DirEx file_mask, return

| Argument | Description |
|---|---|
| File_mask | A string whose value is the DOS file mask that you want to use to list file. File_mask can contain wildcard characters (* and ?). File_mask can contain full or partial file path. |
| Return | A string variable that receives the returned value. |

**Return value:** String. Returns comma-separated list of full file names.

**Usage:** The DirEx statement is similar to the Dir statement. The only difference is that it returns list of full file names including file path while the Dir statement returns list of file names without path. For more information see the topic for the Dirstatement.
To get the list of all files use *.* file mask. If you don't include file path to the file_mask then DirEx statement returns files from the DOS current directory.

**See also:**

Dir
CD

FileSearchEx
FileFindFirst
FTPDir
RemoteDir

# DirExists

**Description:** Tests whether the specified directory exists.

**Syntax:** DirExists dir, status

| Argument | Description |
|---|---|
| Dir | A string whose value is the name of directory that you want to test. |
| Status | A boolean variable that receives the returned status value. |

**Return value:** Boolean. Returns True if the directory exists directory and False otherwise. DirExists statement also returns False if dir is an invalid name or an empty string.

**See also:**

DirDelete
Dir
CD
isDir

# DirRename

**Description:** Renames an existing directory.

**Syntax:** DirRename oldname, newname

| Argument | Description |
|---|---|
| oldname | A string whose value is the name of existing directory that want to rename |
| newname | A string variable whose value is the new directory name |

**Return value:** None

**Usage:** If the newname directory already exists, the statement fails. This statement is provided as a synonym for FileRename

**See also:**

FileRename

FileMove

# DirWaitForUpdate

**Description:** Suspends job execution and then enters an efficient wait state until either a change occurs in the monitored directory or the statement times out.

**Syntax:** DirWaitForUpdate dir, timeout, return_status

| Argument | Description |
|---|---|
| dir | A string whose value is the full name of the directory that you want to monitor. |
| timeout | A number whose value is the maximum time interval within which you expect a new change to occur in the specified dir. Use 0 timeout to allow infinite waiting. |
| return_status | A boolean variable that receives the returned status value. |

**Return value:** Boolean. If the DirWaitForUpdate statement succeeds, the return_status is True, otherwise the return_status value is False. Make sure to specify a valid directory name. Invalid names will cause the statement to return False.

**See also:**

Dir

Wait

FileTime

FileSize

# EOF

**Description:** Checks whether the end of a file (EOF) opened for read operation has been reached.

**Syntax:** EOF filenum, return

| Argument | Description |
|---|---|
| Filenum | The file number previously returned by FileOpen statement |
| return | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE when the end of a file has been reached. The EOF statement returns FALSE until the end of the file has been reached.

**Usage:** Use EOF to avoid the error generated by attempting to read past the end of a file. With files opened for write operations, EOF always returns TRUE.

**See also:**

FileOpen
FileSetPos
FileGetPos
FileSize

# FileAppend

**Description:** Copies an existing file and appends to the end of another file. In other words it concatenates 2 files.

**Syntax:** FileAppend source, destination

| Argument | Description |
|---|---|
| source | A string whose value is the name of an existing file you want to copy |
| destination | A string whose value is the name of an existing or a new file to which you want to append the contents of the source file. |

**Return value:** None

**Usage:** If the destination file already exists, the statement appends contents of the source file to the end of the destination file; otherwise the FileAppend statement creates the new destination file. In the latest case the result is identical to the result that would be produced by the **FileCopy** statement.

**See also:**

FileCopy
FileMove
FileTransfer

# FileExists

**Description:** Reports whether the specified file exists.

**Syntax:** FileExists file, return

| Argument | Description |
|---|---|
| file | A string whose value is the name of a file |
| return | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE if the file exists, FALSE if it does not exist.

**Usage:** If file is locked by another application, causing a sharing violation, FileExists also returns FALSE. File name may include wildcard characters (* and ?).

**See also:**

NotFileExists

# FileCreateTemp

**Description:** Creates an empty temporary file in the default TEMP directory. The name of the file is guaranteed to be unique.

**Syntax:** FileCreateTemp return

| Argument | Description |
|---|---|
| return | A string variable that receives the returned value |

**Return value:** String. Returns name of the created temporary file.

**Usage:** Use FileCreateTemp to create and work with temporary files. After you done with the file processing you should delete the created file using **FileDelete** statement. Repeated failure to delete the file will lead to continues wasting of disk space and potentially slowing down overall system performance after a significant number of files get created and not deleted.

**See also:**

FileOpen
FileWrite
FileSave
FileDelete

# FileClose

**Description:** Closes the file associated with the specified file number. The file number was assigned to the file with the FileOpen statement.

**Syntax:** FileClose filenum

| Argument | Description |
|---|---|
| filenum | The number assigned to the file you want to close. The FileOpen statements returns the file number when it opens the file |

**Return value:** None

**See also:**

FileOpen

# FileCompare

**Description:** Compares contents of two text files, writes differences to the specified output file and then returns number of found differences.

**Syntax:** FileCompare file1, file2, output_file, new_lines1, new_lines2

| Argument | Description |
|---|---|
| file1 | A string whose value is the name of the first file |
| file2 | A string whose value is the name of the second file |
| output_file | A string whose value is the name of the output file which will contain the result of comparision. The output file format is described below. |
| New_lines1 | A numeric variable that receives the returned value for the first file |
| New_lines2 | A numeric variable that receives the returned value for the second file. |

**Return value:** After comparision, the new_lines1 variable containes number of lines that were found in the first file, but not found in the seconf file. The new_lines2 variable containes number of lines that were found in the second file, but not found in the first file. The complete result of file comparion is written to the output_file file.

The output_file file format:
Every line in the output file begins with a 3-character tag following by the original line text.

| Tag | Meaning |
|---|---|
| '  ' (3 spaces) | The identical line is found both files |
| '<! ' | The line is found in the first file only |
| '>! ' | The line found in the second file only |
| '-> ' line1:line2 | The line is found in both files, but in different places. The line1 value is the number of the original line in the first file and line2 value is the number of the original line in the second file. |
| '<- ' line1:line2 | The line is found both files, but in different places. The line1 value is the number of the original line in the first file and line2 value is the number of the original line in the second file. |

**Notes**:

- new_lines1 and new_lines2 are for describe only number of new lines. The compared files can be still different even if both new_lines1 and new_lines2 are equal to 0.
- Don't use FileCompare statement if you only want to compare files without finding all differences. Instead use FileReadAll statement to read the entire first file into a string variable, and then read the entire second file into another variable and then use isEqual statement to compare contents of these variables. The isEqual statement will return either TRUE or FALSE to indicate the difference without going to details.
- The main result of work of the FileCompare statement is the output file whose contens is similar to the results of the UNIX diff command. Diifferences can be easily extracted from the output file using available JAL statements.

**See also:**

FileReadAll
FileSearchEx
FileReplaceEx
isEqual

# FileCopy

**Description:** Copies an existing file to a new file.

**Syntax:** FileCopy source, destination

| Argument | Description |
|---|---|
| source | A string whose value is the name of an existing file you want to copy |
| destination | A string whose value is the name of a new file to which you want to copy the source file |

**Return value:** None

**Usage:** If the destination file already exists, the statement overwrites the existing file.

**See also:**

FileCopyEx
FileAppend
FileMove
FileTransfer

# FileCopyEx

**Description:** Copies existing files to a new location.

**Syntax:** FileCopyEx source, destination, return

| Argument | Description |
|---|---|
| source | A string whose value is the file mask describing existing files that you want to copy. Source file mask can contain wildcard characters (* and ?). |
| destination | A string whose value is the directory name to which you want to copy the source files. |
| return | A numeric variable that receives the returned value. |

**Return value:** Number of files copied. If no files were found to satisfy source file mask, the return value is 0.

**Usage:** If any of the source files already exists in the destination, the statement overwrites the existing file.

**See also:**

- FileCopy
- FileAppend
- FileMoveEx
- FileTransfer

# FileTransfer

**Description:** Copies an existing file from the local drive to a new file on the remote computer.

**Syntax:** FileTransfer host, source, destination

| Argument | Description |
|---|---|
| host | A string whose value is the name of a Remote Agent that is running on the host computer |
| source | A string whose value is the name of an existing file you want to transfer to the host computer |
| destination | A string whose value is the name of a new file which you want to create on the host computer as a copy of the source file |

**Return value:** None

**Usage:** If the destination file already exists, the statement overwrites the existing file. The host name must match the name of an existing Remote Agent profile on the computer initiating the transfer. The transfer time might be lengthily for large files and slow networks. To reduce network traffic and transmission time 24x7 Scheduler compresses source file before sending it to the host. The Remote Agent decompresses received data and stores them in the destination file.

 **Tips:**

- You can transfer multiple files at once. Use comma to separate multiple files in both source and destination parameters. Make sure to specify matching number of files for the source and destination parameters. Sending multiple files in one FileTransfer batch is faster then sending source files one by one using multiple FileTransfer statements.
- You must have sufficient free disk space on both sending and host computers for the temporary files used in compression/decompressions routines.

**See also:**

FileMove
FileCopy

## FileMove

**Description:** Moves an existing file to a new file.

**Syntax:** FileMove source, destination

| Argument | Description |
|---|---|
| source | A string whose value is the name of an existing file you want to move |
| destination | A string whose value is the name of a new file to which you want to move the source file |

**Return value:** None

**Usage:** If the destination file already exists, the statement overwrites the existing file. FileMove first checks the destination file and deletes it when this file exists. Then FileMove copies the source file to the new file. If the copy operation succeeds, FileMove deletes the source file.

**See also:**

FileCopy
FileMoveEx

## FileMoveEx

**Description:** Moves existing files to a new location.

**Syntax:** FileMoveEx source, destination, return

| Argument | Description |
|---|---|
| source | A string whose value is the file mask describing existing files that you want to move. Source file mask can contain wildcard characters (* and ?). |
| destination | A string whose value is the directory name to which you want to move the source files. |
| return | A numeric variable that receives the returned value. |

**Return value:** Number of files moved. If no files were found to satisfy source file mask, the return value is 0.

**Usage:** If any of the source files already exists in the destination, the statement overwrites the existing file.

**See also:**

FileMove
FileCopyEx
FileTransfer

## FileDate

**Description:** Reports the date that a file was last modified.

**Syntax:** FileDate file, return

| Argument | Description |
|---|---|
| file | A string whose value is the name of a file |
| return | A date variable that receives the returned value |

**Return value:** Date. Returns the date that a file was last modified..

**See also:**

FileTime
MakeDateTime

## FileTime

**Description:** Reports the time that a file was last modified.

**Syntax:** FileTime file, return

| Argument | Description |
|---|---|
| file | A string whose value is the name of a file |
| return | A time variable that receives the returned value |

**Return value:** Time. Returns the time that a file was last modified.

**See also:**

FileDate
MakeDateTime

# FileDelete

**Description:** Deletes the specified file.

**Syntax:** FileDelete file

| Argument | Description |
|---|---|
| file | A string whose value is the name of a file you want delete |

**Return value:** None

**See also:**

FileDeleteEx

# FileDeleteEx

**Description:** Deletes existing files from the specified location.

**Syntax:** FileDeleteEx mask, return

| Argument | Description |
|---|---|
| mask | A string whose value is the file mask describing existing files that you want to delete. File mask can contain can contain wildcard characters (* and ?). |
| return | A numeric variable that receives the returned value. |

**Return value:** Number of files deleted. If no files were found to satisfy file mask, the return value is 0.

**See also:**

FileDelete

FileMoveEx

# FileFindFirst

**Description:** Searches a directory for a file whose name matches the specified filename. FileFindFirst examines subdirectory names as well as filenames.

**Syntax:** FileFindFirst filename, firstfile, return

| Argument | Description |
|---|---|
| filename | A string whose value specifies a valid directory or path and filename, which can contain wildcard characters (* and ?). |
| firstfile | A string variable that receives the name of a file whose name matches the specified filename |
| return | A boolean variable that receives the success of the operation. If the search was successful return receives TRUE, otherwise it receives FALSE. |

**Return value:** This statements returns two values: string and boolean. If the statement succeeds, the string variable firstfile receives the name of the first found file.

**Usage:** If the statement succeeds, you can use the FileFindNext statement to search for other files that match the same pattern. You may wont to call FileFindNext in a loop in order to find all files that match the same pattern.

**See also:**

FileFindNext
Dir
DirEx
SubDir
isDir
FileExists
LoopWhile
LoopUntil

# FileFindNext

**Description:** Searches a directory for a file whose name matches the specified filename. FileFindNext examines subdirectory names as well as filenames.

**Syntax:** FileFindNext nextfile, return

| Argument | Description |
|---|---|
| nextfile | A string variable that receives the name of a file whose name matches the mask specified in the previous call to the FindFileFirst statement |
| return | A boolean variable that receives the success of the operation. If the search was successful return receives TRUE, otherwise it receives FALSE. |

**Return value:** This statements returns two values: string and boolean. If the statement succeeds, the string variable nextfile receives the name of the next found file.

**Usage:** You use the FileFindNext statement to continue a file search from a previous call to the FindFileFirst statement. You may wont to call FileFindNext in a loop in order to find all files that match the same pattern.

**See also:**

FileFindFirst
Dir
DirEx
SubDir
isDir
FileExists
LoopWhile
LoopUntil

# FileSearchEx

**Description:** Searches for files that contain the specified text.

**Syntax:** FileSearchEx dir, file_mask, find_string, search_subdir, return

| Argument | Description |
|---|---|
| Dir | A string whose value is the full name of the directory where you want to search for files |
| File_mask | A string whose value is the file mask describing existing files that you want to search. File_mask can contain wildcard characters (* and ?). |
| Find_string | A string whose value is the text contained in file(s) to be found |
| Search_subdir | A boolean that indicates whether you want to search in the subdirectories starting with dir. Specify TRUE to search in the subdirectories, or FALSE to search only in the directory specified by dir. |
| Return | A string variable that receives the returned value. |

**Return value:** String. Returns comma-separated list of files that contain the specified find_string.

**Usage:**
FileSearchEx statement should be used for searching in text files only, such as .TXT, .HTM, .LOG, and so on. It may produce incorrect results when searching in binary files.

You may want to use FileSearchEx to quickly scan various log files for common error messages.

**See also:**

FileReplaceEx
FileFindFirst
Pos
InStr

# FileReplaceEx

**Description:** Searches and replaces in files.

**Syntax:** FileReplaceEx dir, file_mask, find_string, replace_string, search_subdir, return

| Argument | Description |
|---|---|
| Dir | A string whose value is the full name of the directory where you want to search for files |
| File_mask | A string whose value is the file mask describing existing files that you want to search and replace. File_mask can contain wildcard characters (* and ?). |
| Find_string | A string whose value is the text contained in file(s) to be found |
| Replace_string | A string whose value is the text that is used to replace all occurrences of the Find_string |
| Search_subdir | A boolean that indicates whether you want to search and replace in the subdirectories starting with dir. Specify TRUE to search in the subdirectories, or FALSE to search only in the directory specified by dir. |
| Return | A numeric variable that receives the returned value. |

**Return value:** Number. Returns total number of replacements made in all files that were found to contain the specified find_string.

**Usage:** FileReplaceEx statement should be used for searching and replacing in text files only, such as .TXT, .HTM, .LOG, and so on. It may corrupt binary files.

 **Tip:**
FileReplaceEx statement can be used to quickly convert files from UNIX format to DOS/Windows format and vise versa.
Note that UNIX files use NL character (ASCII code 10) as a end of line markers where DOS files use CR/NL pair of characters (ASCII codes 13 and 10) for this purpose. Use FileReplaceEx statement to convert files after FTP downloading from UNIX or before FTP uploading to UNIX hosts.
Examples:

```
UNIX to DOS: FileReplaceEx  "*.htm", "\n", "\r\n", False, count
DOS to UNIX: FileReplaceEx  "*.htm", "\r\n", "\n", False, count
```

**See also:**

- FileSearchEx
- FileFindFirst
- FileConvert
- Replace

# FileConvert

**Description:** Converts the specified file by replacing some symbols.

**Syntax:** FileConvert file, find_table, replace_table, return

| Argument | Description |
|---|---|
| File | A string whose value is the full name of the file in which you want to replace symbols from Source_list string with matching symbols from the Transform_list string. |
| Source_list | A comma-separated string containing ASCII codes of symbols to be converted (replaced) |
| Transform_list | A comma-separated string containing ASCII codes of symbols to be used for conversion. |
| Return | A numeric variable that receives the returned value. |

**Return value:** Number. Returns number of replacements made in the file.

**Usage:** Number of symbols in the Transform_list string must match number of symbols in the Source_list sring. The FileConvert statement scans the specified file for symbols included into Source_list. For every found occurrence, the FileConvert statement replaces the found symbol with the corresponding symbol from the Transform_list. This allows you to perform multi-symbol "replace all" operation on a file.

FileConvert statement can process both text and binary files.

**Tip:**
You may use FileConvert statement to convert mainframe files using EBCDIC code table to files using ASCII code table. Translations between code sets can be tricky. There are a variety of problems that may occur. A problem related to translation is record format. In mainframe world, records exist as fixed-length, variable-length and string. There are no delimiters. All file types may contain text or binary data. When sent via FTP, however, there are no fixed-length records. Also, EBCDIC records are always ended by a new line (NL) character (X'15') and ASCII records are ended with a carriage return, line feed pair (CR LF). These characters cannot be embedded in the records. Binary files are treated as a single string of bytes and such files have no record structure. Thus, choice of translation affects the record format of the file. FileConvert statement provides powerful and yet simple options for translating between ASCII and EBCDIC. 24x7 Scheduler is shipped with the Example Jobs database that includes two user-defined JAL statements for performing EBCDIC to ASCII and ASCII to EBCDIC file conversions using common translation tables presented as translation lists.
You can customize translations lists in these example statements to make them fit your unique requirements.

Examples:

This statement replaces ASCII symbols 0 and 7 with a space (ASCII code 32) and a tab (ASCII code 8) symbols
FileConvert "c:\ftp\myfile.txt", "0,7", "32,8", count

This statement replaces ASCII symbol 0 with a space (ASCII code 32)
FileConvert "c:\ftp\myfile.txt", "0", "32", count

### Common Translation Tables

### EBCDIC to ASCII

```
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
0 00 01 02 03 DC 09 C3 7F CA B2 D5 0B 0C 0D 0E 0F
1 10 11 12 13 DB DA 08 C1 18 19 C8 F2 1C 1D 1E 1F
2 C4 B3 C0 D9 BF 0A 17 1B B4 C2 C5 B0 B1 05 06 07
3 CD BA 16 BC BB C9 CC 04 B9 CB CE DF 14 15 FE 1A
4 20 FF 83 84 85 A0 C6 86 87 A4 BD 2E 3C 28 2B 7C
5 26 82 88 89 8A A1 8C 8B 8D E1 21 24 2A 29 3B AA
6 2D 2F B6 8E B7 B5 C7 8F 80 A5 DD 2C 25 5F 3E 3F
```

```
7 9B 90 D2 D3 D4 D6 D7 D8 DE 60 3A 23 40 27 3D 22
8 9D 61 62 63 64 65 66 67 68 69 AE AF D0 EC E7 F1
9 F8 6A 6B 6C 6D 6E 6F 70 71 72 A6 A7 91 F7 92 CF
a E6 7E 73 74 75 76 77 78 79 7A AD A8 D1 ED E8 A9
b 5E 9C BE FA B8 F5 F4 AC AB F3 5B 5D EE F9 EF 9E
c 7B 41 42 43 44 45 46 47 48 49 F0 93 94 95 A2 E4
d 7D 4A 4B 4C 4D 4E 4F 50 51 52 FB 96 81 97 A3 98
e 5C F6 53 54 55 56 57 58 59 5A FD E2 99 E3 E0 E5
f 30 31 32 33 34 35 36 37 38 39 FC EA 9A EB E9 9F
```

### ASCII to EBCDIC

```
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
0 00 01 02 03 37 2D 2E 2F 16 05 25 0B 0C 0D 0E 0F
1 10 11 12 13 3C 3D 32 26 18 19 3F 27 1C 1D 1E 1F
2 40 5A 7F 7B 5B 6C 50 7D 4D 5D 5C 4E 6B 60 4B 61
3 F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 7A 5E 4C 7E 6E 6F
4 7C C1 C2 C3 C4 C5 C6 C7 C8 C9 D1 D2 D3 D4 D5 D6
5 D7 D8 D9 E2 E3 E4 E5 E6 E7 E8 E9 BA E0 BB B0 6D
6 79 81 82 83 84 85 86 87 88 89 91 92 93 94 95 96
7 97 98 99 A2 A3 A4 A5 A6 A7 A8 A9 C0 4F D0 A1 07
8 68 DC 51 42 43 44 47 48 52 53 54 57 56 58 63 67
9 71 9C 9E CB CC CD DB DD DF EC FC 70 B1 80 BF FF
a 45 55 CE DE 49 69 9A 9B AB AF 5F B8 B7 AA 8A 8B
b 2B 2C 09 21 28 65 62 64 B4 38 31 34 33 4A B2 24
c 22 17 29 06 20 2A 46 66 1A 35 08 39 36 30 3A 9F
d 8C AC 72 73 74 0A 75 76 77 23 15 14 04 6A 78 3B
e EE 59 EB ED CF EF A0 8E AE FE FB FD 8D AD BC BE
f CA 8F 1B B9 B6 B5 E1 9D 90 BD B3 DA FA EA 3E 41
```

**See also:**

FileSearchEx
FileReplaceEx
Replace
FTP statements
Telnet statements

# FileGetAttr

**Description:** Reports file attributes for the specified file or directory.

**Syntax:** FileGetAttr file, return

| Argument | Description |
|---|---|
| file | A string variable that is the file or directory name you want to query |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns a number representing the attributes of a file, directory, or folder.

**Usage:** The value returned by FileGetAttr is the sum of any following attribute values:

| Value | Description |
|---|---|
| 0 | Normal |
| 1 | Read-only |
| 2 | Hidden |
| 4 | System |
| 16 | Directory or folder |
| 32 | File has changed since last backup |

To determine which attributes are set, use the BitwiseAnd statement to perform a bitwise comparison of the value returned by the FileGetAttr statement and the value of the individual file attribute you want. If the result is not zero, that attribute is set for the named file.

**See also:**

FileSetAttr

Bitwise statements

# FileSetAttr

**Description:** Sets file attributes for the specified file or directory.

**Syntax:** FileSetAttr file, attr

| Argument | Description |
|---|---|
| File | A string variable that is the file or directory name you want to update |
| Attr | A numeric variable that is the new file attributes |

**Return value:** None

**Usage:** The attr value for the FileSetAttr is the sum of any following attribute values:

| Value | Description |
|---|---|
| 0 | Normal |
| 1 | Read-only |
| 2 | Hidden |
| 4 | System |
| 16 | Directory or folder |
| 32 | File has changed since last backup |

**Note:**
A run-time error will occur if you try to change attributes of an open file.

**See also:**

FileGetAttr

FileSetAttrEx

# FileSetAttrEx

**Description:** Sets file attributes for all files that match the specified file mask.

**Syntax:** FileSetAttrEx file, attr, count

| Argument | Description |
|---|---|
| file | A string variable that is the file mask you want to use for searching files whose attributes will be updated. You can use standard * and ? wildcard characters for the mask. |
| attr | A numeric variable that is the new file attributes |
| count | A numeric variable that receives the returned value |

**Return value:** Number. Returns the number of files whose attributes were changed.

**Usage:** The attr value for the FileSetAttrEx is the sum of any following attribute values:

| Value | Description |
|---|---|
| 0 | Normal |
| 1 | Read-only |
| 2 | Hidden |
| 4 | System |
| 16 | Directory or folder |
| 32 | File has changed since last backup |

**Note:**

- A run-time error occurs if you try to set the attributes of an open file.
- FileSetAttrEx can update multiple files at once but it is not recursive. If you want to update files in subdirectories you need to execute FileSetAttrEx separately for each subdirectory.

**See also:**

FileGetAttr
FileSetAttr

# FileGetPos

**Description:** Reports current position in the specified file previously opened by FileOpen statement.

**Syntax:** FileGetPos filenum, pos

| Argument | Description |
|---|---|
| Filenum | The file number previously assigned to the file when it was opened by FileOpen statement |
| Pos | A numeric variable that receives the returned value |

**Return value:** Number. Returns the file position after the read/write operation or zero if no operation has been performed after file opening.

**Note:**
FileGetPos always returns position from the beginning of the file.

**See also:**

FileSetPos
FileOpen

# FileSetPos

**Description:** Moves the file pointer to the specified position in a file previously opened by FileOpen statement. The file pointer is the position in the file at which the next read or write begins.

**Syntax:** FileSetPos filenum, origin, pos

| Argument | Description |
|---|---|
| Filenum | The file number previously assigned to the file when it was opened by FileOpen statement |
| Pos | A number whose value is the new position of the file pointer relative to the position specified in origin, in bytes |
| origin | A string constant whose value specifies from where you want to set then position. Values are:<br>• "START" — At the beginning of the file<br>• "CURRENT" — At the current position<br>• "END" — At the end of the file |

**Return value:** None.

**Usage:** The file must be opened previously using FileOpen statement.

**See also:**

FileGetPos
FileOpen

# FileOpen

**Description:** Opens the specified file for reading or writing and assigns it a unique file number. You use this number to identify the file when you read, write, or close the file. The arguments filemode, fileaccess, filelock, and append determine the mode in which the file is opened. If the file doesn't exist, a new file is created.

**Syntax:** FileOpen filename, filemode, fileaccess, append ,return

| Argument | Description |
|---|---|
| filename | A string whose value is the name of the file you want to open. If filename is not on the operating system's search path, you must enter the fully qualified name |
| filemode | A string constant whose value specifies how the end of a file read or file write operation is determined. Values are:<br>• "LineMode" — Read or write the file a line at a time<br>• "StreamMode" — Read the file in 32K chunks. For more information, see Usage below |
| fileaccess | A string constant whose value specifies whether the file is opened for reading or writing. Values are:<br>• "Read" — Read-only access<br>• "Write" — Write-only access |
| append | A boolean whose value specifies whether existing data in the file is overwritten when file is opened for write operation. Values are:<br>• True — Write data to the end of the file<br>• False — Replace all existing data in the file<br>append is ignored if the fileaccess argument is "Read" |
| return | A numeric variable that receives the returned file number. |

**Return value:** Number. Returns the file number assigned to filename.

**Usage:** When a file has been opened in "LineMode" , each call to the FileRead statement reads until it encounters a carriage return (CR), linefeed (LF), or end-of-file mark (EOF). Each call to the FileWrite adds a CR and LF at the end of each string it writes.
When a file has been opened in "StreamMode" , a call to FileRead reads the whole file (until it encounters an EOF) or 32,765 bytes, whichever is less. FileWrite writes a maximum of 32,765 bytes in a single call and does not add CR and LF characters.

**Note:**
If the 24x7 Scheduler doesn't find the file, it creates a new file, giving it the specified name.

**See also:**

FileClose
FileRead
FileWrite

# FilePrint

**Description:** Sends the specified file to the default printer

**Syntax:** FilePrint filename

| Argument | Description |
|---|---|
| filename | A string whose value is the name of the file you want to print. If filename is not on the operating system's search |

| | |
|---|---|
| | path, you must enter the fully qualified name |

**Return value:** None.

**See also:**

PrinterSetDefault

# FileRead

**Description:** Reads data from the file associated with the specified file number, which was assigned to the file with the FileOpen statement.

**Syntax:** FileRead filenum, return

| Argument | Description |
|---|---|
| filenum | The file number previously assigned to the file when it was opened by FileOpen statement |
| return | A string variable into which you want to read the data |

**Return value:** String.
If the file is opened in Line mode, FileRead reads a line of the file (that is, until it encounters a CR, LF, or EOF). It stores the contents of the line in the specified variable, skips the line-end characters, and positions the file pointer at the beginning of the next line.

If the file was opened in Stream mode, FileRead reads to the end of the file or the next 32,765 bytes, whichever is shorter. FileRead begins reading at the file pointer, which is positioned at the beginning of the file when the file is opened for reading. If the file is longer than 32,765 bytes, FileRead automatically positions the pointer after each read operation so that it is ready to read the next chunk of data.

FileRead can read a maximum of 32,765 characters at a time. Therefore, before calling the FileRead function, call the FileSize statement to check the file size. If your system has file sharing or security restrictions, you may need to call FileSize before you call FileOpen.

An end-of-file mark is a NULL character (ASCII value 0). Therefore, if the file being read contains null characters, FileRead will stop reading at the first null character, interpreting it as the end of the file.

**Usage:** The file must be opened previously using FileOpen statement.

**See also:**

FileSize

FileOpen

# FileReadAll

**Description:** Loads data from the specified file.

**Syntax:** FileReadAll file, return

| Argument | Description |
|---|---|
| file | A string whose value is the name of the existing file that you want to read |
| return | A string variable into which you want to read the data |

**Return value:** String. Loads entire file specified by file into a string variable specified by return.

**See also:**

FileRead
FileSave
FileCopy

# FileReadLine

**Description:** Reads specified line from a ASCII file.

**Syntax:** FileReadLine file, line, return

| Argument | Description |
|---|---|
| file | A string whose value is the name of the existing file that you want to read |
| line | A number whose value is the number of the line that want to read from the file |
| return | A string variable into which you want to read the data |

**Return value:** String. Reads specified line from file.

**Usage:** FileReadLine opens file for reading in the Line Mode, reads the file until it reaches the specified line, then closes the file. This statement makes sense only for ASCII (text) files.

**See also:**

FileRead
FileReadAll

# FileRename

**Description:** Renames an existing file.

**Syntax:** FileRename oldname, newname

| Argument | Description |
|---|---|
| oldname | A string whose value is the name of existing file or directory that want to rename |
| newname | A string variable whose value is the new file name |

**Return value:** None

**Usage:** If the newname file already exists, the statement fails. You can use FileMove to override an existing file. Both oldname and newname can include path to the file.

**See also:**

FileCopy

FileMove

# FileSplitName

**Description:** Splits up a full file name into two components consisting of path, file name with extension

**Syntax:** FileSplitName fullname, filepath, filename

| Argument | Description |
|---|---|
| fullname | A string whose value is the full file name that may include file path and file extension |
| filepath | A string variable whose value is the returned file path including drive letter and directory name |
| filename | A string variable whose value is the returned file name including file extension |

**Return value:** String and String

**Usage:** The specified file does not have to exist.

**See also:**

FileCopy

FileMove

# FileSave

**Description:** Saves data in the specified file.

**Syntax:** FileSave file, text

| Argument | Description |
|---|---|
| file | A string variable whose value is the name of the file into which you want to save the text |
| text | A string whose value is the data want to save in the file |

**Return value:** None.

**Usage:** If the file already exists then FileSave overwrites it.

**See also:**

FileRead
FileExists
FileCopy

# FileSize

**Description:** Reports the length of a file in bytes.

**Syntax:** FileSize file, return

| Argument | Description |
|---|---|
| file | A string whose value is the name of the file for which you want to know the length. If filename is not on the current application library search path, you must specify the fully qualified name |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns the length in bytes of the file identified by file.

# FileWrite

**Description:** Write data to the file associated with the specified file number, which was assigned to the file with the FileOpen statement.

**Syntax:** FileWrite filenum, s

| Argument | Description |
|---|---|
| filenum | The file number previously assigned to the file when it was opened by FileOpen statement |
| s | A string variable, which you want to save to the file associated with filenum |

**Return value:** None.

**Usage:**
FileWrite writes its data at the position identified by the file pointer. If the file was opened by FileOpen with the writemode argument set to “Replace" , the file pointer is initially at the beginning of the file. After each call to FileWrite, the pointer is immediately after the last write. If the file was opened with the writemode argument set to “Append" , the file pointer is initially at the end of the file and moves to the end of the file after each write.

FileWrite sets the file pointer following the last character written. If the file was opened in “LineMode" , FileWrite writes a carriage return (CR) and linefeed (LF) after the last character in variable and places the file pointer after the CR and LF.
FileWrite can write only 32,766 bytes at a time, which includes the string terminator character. If the length of variable s exceeds 32,765, FileWrite writes the first 32,765 characters only.

**Usage:** The file must be opened previously using FileOpen statement.

**See also:**

FileRead
FileOpen

# FileZip

**Description:** Compresses existing files into a new ZIP file. Produced ZIP files are compatible with PKZIP, Winzip, InfoZip, and most other zipping utilities. PKZIP utility is not required for zipping operations.

**Syntax:** FileZip destination, source

| Argument | Description |
|---|---|
| destination | A string whose value is the name of a ZIP file into which you want to archive the source file |
| source | A string whose value is the comma-separated list of existing files you want to zip. |

**Return value:** None

**Usage:** If the destination file already exists, FileZip overwrites the existing file.

**Tips:**

- Source file names may include wildcard characters. Multiple files and wildcards can be specified in a single FileZip operation.
- Use FileZipEx statement if you want to recursively zip files in subdirectories.

**See also:**

FileZipEx
FileUnzip
FileDeleteEx
Dir
CD

# FileZipEx

**Description:** Compresses existing files into a new ZIP file. Produced ZIP files are compatible with PKZIP, Winzip, InfoZip, and most other zipping utilities. PKZIP utility is not required for zipping operations.

**Syntax:** FileZipEx destination, source, recursive, save_path, base_dir

| Argument | Description |
|---|---|
| destination | A string whose value is the name of a ZIP file into which you want to archive the source file(s) |
| source | A string whose value is the comma-separated list of existing files you want to zip. |
| recursive | A boolean whose value controls whether the zip operation should recursively process file in subdirectories. |
| save_path | A boolean whose value controls whether the path to zipped files should be stored within archive. In WinZip and other popular compression utilities this option is often called "save extra-folder info." |
| base_dir | A string whose value defines which part of the file path should be stored in archive. This option is ignored if save_path is set to False. For example, if you want to zip all files in c:\dir1\dir2*.* and all subdirectories but save "extra folder info" beginning with dir2 you should specify c:\dir1 as base_dir. If There exist files<br>c:\dir1\dir2\file1,<br>c:\dir1\dir2\file2,<br>c:\dir1\dir2\dir3\file3<br>then within the resulting archive they would be stored as<br>dir2\file1,<br>dir2\file2,<br>dir2\dir3\file3 |

**Return value:** None

**Usage:** If the destination file already exists, FileZipEx overwrites the existing file. FileZipEx behaves exactly as the **FileZip** statement when both recursive and save_path options are set to False.

 **Tip:** Source file names may include wildcard characters. Multiple files and wildcards can be specified in a single FileZipEx operation. Example:

```
FileZipEx "c:\\backup\\backup.zip", "c:\\dir1\\*.txt, c:\\dir1\\*.htm", True, True, "c:\\"
```

**See also:**

FileZip
FileUnzip
FileDeleteEx
Dir
CD

## FileUnzip

**Description:** Decompresses files from an existing ZIP file. (PKUNZIP utility is not required for unzipping operations)

**Syntax:** FileUnzip source, destination

| Argument | Description |
|---|---|
| source | A string whose value is the name of existing ZIP file. |
| destination | A string whose value is the name of existing directory to which all archived files will be extracted from the source file. |

**Return value:** None

**Usage:** If the destination directory does not exist, the statement fails. Extracted files may overwrite existing files with the same names in the destination directory.

**See also:**

FileZip
Dir
FileDeleteEX

## IniFileGetKey

**Description:** Obtains the string value of a setting in the profile file or system registry. A profile file usually has .INI extension.

**Syntax:** IniFileGetKey inifile, section, key, return

| Argument | Description |
|---|---|
| inifile | A string whose value is the name of the profile file. If you do not specify a full path, IniFileGetKey uses the operating system's standard file search order to find the file |
| section | A string whose value is the name of a group of related values in the profile file. In the file, section names are in square brackets. Do not include the brackets in section. Section is not case-sensitive |
| key | A string specifying the setting name in section whose value you want. The setting name is followed by an equal sign in the file. Do not include the equal sign in key. Key is not case-sensitive |
| return | A string variable, which receives the returned value |

**Return value:** String, with a maximum length of 4096 characters. Returns the string from key within section within inifile. If inifile is not found, section is not found in inifile, or key is not found in section, IniFileGetKey returns the empty string (""). If an error occurs, it returns the empty string ("").

**Usage:** Use IniFileGetKey to get configuration settings from a profile file. You can use IniFileSetKey statement to change configuration of the application that owns this profile file. Before you make changes, you can use IniFileGetKey to obtain the original settings so you can restore them after or during the application run.

IniFileGetKey can also be used to obtain configuration settings from the Windows system registry. To obtain information from the registry instead of from an initialization file:

- On Windows NT (NT/2000/XP/2003/VISTA/2008/7), create a new folder called INIFILEMAPPING at the following location: hkey_current_user\software\microsoft\windows nt\current version To override the WIN.INI file, create a key in the new folder called WIN.INI with the following value: #usr:software\microsoft\windows nt\current version\extensions
- On Windows 95/98/Me, substitute windows for windows nt in both paths (see above).

This change will tell Windows to override initialization files, so that you can use these initialization files to obtain registry settings.

**See also:**

IniFileSetKey
RegistryGetKey
RegistrySetKey

# IniFileSetKey

**Description:** Changes the string value of a setting in the profile file or system registry. A profile file usually has .INI extension.

**Syntax:** IniFileSetKey inifile, section, key, newvalue

| Argument | Description |
|---|---|
| inifile | A string whose value is the name of the profile file. If you do not specify a full path, IniFileSetKey uses the operating system's standard file search order to find the file |

| | |
|---|---|
| section | A string whose value is the name of a group of related values in the profile file. In the file, section names are in square brackets. Do not include the brackets in section. Section is not case-sensitive |
| key | A string specifying the setting name in section whose value you want to change. The setting name is followed by an equal sign in the file. Do not include the equal sign in key. Key is not case-sensitive |
| newvalue | A string whose value is the value you want to specify for key. The value can be up to 4096 characters long. |

**Return value:** None.

**Usage:** IniFileSetKey changes the value for key within section within inifile. If inifile is not found, section is not found in inifile, or key is not found in section, an error occurs.

Use IniFileSetKey to change configuration settings stored within profile file. You can use IniFileSetKey statement to change configuration of the application that owns this profile file. Before you make changes, you can use IniFileGetKey to obtain the original settings so you can restore them after or during the application run.

IniFileSetKey can also be used to change configuration settings in the Windows system registry. To change information in the registry instead of changing an initialization file:

- On Windows NT (NT/2000/XP/2003/VISTA/2008/7), create a new folder called INIFILEMAPPING at the following location: hkey_current_user\software\microsoft\windows nt\current version To override the WIN.INI file, create a key in the new folder called WIN.INI with the following value: #usr:software\microsoft\windows nt\current version\extensions
- On Windows 95/98/Me, substitute windows for windows nt in both paths (see above).

This change will tell Windows to override initialization files, so that you can use these initialization files to obtain registry settings.

**See also:**

IniFileGetKey
RegistryGetKey
RegistrySetKey

# isDir

**Description:** Tests whether the specified path is a name of a valid existing directory.

**Syntax:** isDir dir, status

| Argument | Description |
|---|---|
| Dir | A string whose value is the name of directory that you want to test. |
| Status | A boolean variable that receives the returned status value. |

**Return value:** Boolean. Returns True if the name points to an existing directory and False otherwise.

**See also:**

DirDelete
Dir
CD

## NotFileExists

**Description:** Reports whether the specified file does not exist.

**Syntax:** NotFileExists file, return

| Argument | Description |
|---|---|
| file | A string whose value is the name of a file |
| Return | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE if the file does not exist, FALSE if it exists.

**Usage:** If file is locked by another application, causing a sharing violation, NotFileExists also returns TRUE. File name may include wildcard characters (* and ?).

**See also:**

FileExists

## RemoteDir

**Description:** Returns comma-separated list of files in the specified directory on the specified remote computer running 24x7 Remote Agent

**Syntax:** RemoteDir agent, file_mask, return

| Argument | Description |
|---|---|
| Agent | A string whose value is the name of the Remote Agent as it is specified in the Remote Agent profile. |
| File_mask | A string whose value is the file mask that you want to use to list file. File_mask can contain wildcard characters (* and ?). File_mask can contain full or partial file path. |
| Return | A string variable that receives the returned value. |

**Return value:** String. Returns comma-separated list of files.

**Usage:** The RemoteDir statement is equivalent to DOS *dir* command executed on the remote computer hosting 24x7 Remote Agent specified by Agent.
If you don't include file path to the file_mask then RemoteDir statement returns files from the Agent's computer current directory.

**See also:**
FileTransfer
Dir
FTPDir

# File replication and synhronization statements

## CompareFTPDir

**Description:** Compares files in two directories residing on local and remote computers using FTP protocol.

**Syntax:** CompareFTPDir server, user, password, local_dir, remote_dir, name_comparison, local_list, remote_list

| Argument | Description |
|---|---|
| Server | A string whose value is host name of an FTP server (for example, ftp.microsoft.com) or the IP number of the site in ASCII dotted-decimal format (for example, 11.0.1.45) |
| User | A string whose value is the name of the user to log on |
| Password | A string whose value is the password to use to log on |
| Local_dir | A string whose value is the name of the local directory containing files that you want to compare |
| Remote_dir | A string whose value is the name of the remote directory on FTP server containing files that you want to compare. |
| Name_comparison | A boolean whose value should be TRUE if you want to compare files by name only, and FALSE if you want to compare them by name and date of the last modification. |
| Local_list | A string variable that receives the list of files from the Local_dir directory. This list contains only files that are different from files in the Remote_dir directory. |
| Remote_list | A string variable that receives the list of files from the Target_dir directory. This list contains only files that are different from files in the Remote_dir |

| | |
|---|---|
| | directory. |

**Return value:** Two String values are returned.

**Usage:** Use CompareFTPDir statement to automate various file comparison and synchronization processes.

By default 24x7 Scheduler uses standard non-secure FTP protocol for all FTP operations. If you are working with a secure FTP server (SFTP server) use **FTPConfig** statement to switch FTP protocol.

By default 24x7 Scheduler performs all FTP operations using default port for FTP servers. To specify a different port number use **FTPConfig** statement.

File names are case-sensitive on all FTP server platforms.

**See also:**
SyncFTPDir
CompareLocalDir
CompareRemoteDir
FTPGetFile
FTPPutFile

# CompareLocalDir

**Description:** Compares files in two local or shared network directories.

**Syntax:** CompareLocalDir source_dir, target_dir, name_comparison, source_list, target_list

| Argument | Description |
|---|---|
| Source_dir | A string whose value is the name of the source directory containing files that you want to compare |
| Target_dir | A string whose value is the name of the target directory |
| Name_comparison | A boolean whose value should be TRUE if you want to compare files by name only, and FALSE if you want to compare them by name and date of the last modification. |
| Source_list | A string variable that receives the list of files from the Source_dir directory. This list contains only files that are different from files in the Target_dir directory. |
| Remote_dir | A string whose value is the name of the remote directory on FTP server containing files that you want to compare. |

**Return value:** Two String values are returned.

**Usage:** Use CompareLocalDir statement to automate various file comparison and synchronization processes.

**See also:**
SyncLocalDir
CompareFTPDir

CompareRemoteDir
FileCopyEx
FileMoveEx

# CompareRemoteDir

**Description:** Compares files in two directories residing on local and remote computers provided 24x7 Remote Agent is running at a given remote computer.

**Syntax:** CompareRemoteDir agent, local_dir, remote_dir, name_comparison, local_list, remote_list

| Argument | Description |
|---|---|
| Agent | A string whose value is the name of 24x7 Remote Agent or 24x7 Master Scheduler as it is specified in the Remote Agent profile. |
| Local_dir | A string whose value is the full name of the local directory containing files that you want to compare |
| Remote_dir | A string whose value is the full name of the remote directory containing files that you want to compare. |
| Name_comparison | A boolean whose value should be TRUE if you want to compare files by name only, and FALSE if you want to compare them by name and date of the last modification. |
| Local_list | A string variable that receives the list of files from the Local_dir directory. This list contains only files that are different from files in the Remote_dir directory. |
| Remote_list | A string variable that receives the list of files from the Target_dir directory. This list contains only files that are different from files in the Remote_dir directory. |

**Return value:** Two String values are returned.

**Usage:** Use CompareRemoteDir statement to automate various file comparison and synchronization processes.

**See also:**
SyncRemoteDir
CompareLocalDir
CompareFTPDir
FileTransferEx

# SyncFTPDir

**Description:** Synchronizes and replicates files across two directories residing on local and remote computers using FTP protocol.

**Syntax:** SyncFTPDir master, server, user, password, source_dir, target_dir, add_new, delete_missing, update_old, subdir

| Argument | Description |
|---|---|
| Master | A string whose value instructs 24x7 Schedule which computer is the "master" computer containing files and subdirectories that you want to replicate.<br>The following values are supported:<br>• "REMOTE" - the remote FTP server is "master"<br>• "LOCAL" the local computer is "master" |
| Server | A string whose value is host name of an FTP server (for example, ftp.microsoft.com) or the IP number of the site in ASCII dotted-decimal format (for example, 11.0.1.45) |
| User | A string whose value is the name of the user to log on |
| Password | A string whose value is the password to use to log on |
| Source_dir | A string whose value is the name of the "master" directory containing files and subdirectories that you want to replicate |
| Target_dir | A string whose value is the name of the target directory to which .files and subdirectories are replicated |
| Add_new | A boolean whose value should be TRUE if you want to replicate files that exist in the source_dir, and FALSE otherwise |
| Delete_missing | A boolean whose value should be TRUE if you want to delete from the target_dir directory these files that target_dir but do not exist in the source_dir, and FALSE otherwise |
| Update_old | A boolean whose value should be TRUE if you want to update older versions of files in the target_dir directory, and FALSE otherwise. If file is considered as old if it exists in both target_dir and source_dir directories and target_dir version of that files has a time older than the version from the source_dir directory. |
| Subdir | A boolean whose value should be TRUE if you want to update recursive subdirectories. Note that if you enable recursion then all other replication options described above apply to all subdirectories of all nesting levels starting with the Source_dir. |

**Return value:** None.

**Usage:** Use SyncFTPDir statement to automate synchronization and replication for a group of files residing on different computers.
The directory you are copying files from is also known as the **master directory** or **primary site**-replication

24x7 Scheduler logs all file synchronization and replication details to the SYNC.LOG file in the 24x7 Scheduler's installation directory.

By default 24x7 Scheduler uses standard non-secure FTP protocol for all FTP operations. If you are working with a secure FTP server (SFTP server) use **FTPConfig** statement to switch FTP protocol.

By default 24x7 Scheduler performs all FTP operations using default port for FTP servers. To specify a different port number use **FTPConfig** statement.

File names are case-sensitive on all FTP server platforms.

**Important Notes:**

- Because the standard FTP protocol lacks file date/time manipulations, all files uploaded to a FTP server always have system date/time on the FTP server computer (date/time when they were uploaded).
- Read **File Caching Internet Options** topic if you experience various "cannot create file/directory because it already exists" errors or files do not get updated during FTP downloads.

**Tips:**

- To update only outdated files set Update_old to TRUE and set both Add_new and Delete_missing to FALSE.
- To perform full 2 way file synchronization between directory <one> residing on the remote computer and directory <two> residing on the local computer:
  First run SyncFTPDir using "REMOTE" for the Master with Delete_missing set to FALSE and everything else set to TRUE. Repeat SyncFTPDir using "LOCAL" for the Master with Delete_missing set to FALSE and everything else set to TRUE.

**See also:**

SyncRemoteDir
SyncLocalDir
Other FTP statements

# SyncLocalDir

**Description:** Synchronizes and replicates files across two local or shared network directories.

**Syntax:** SyncLocalDir source_dir, target_dir, add_new, delete_missing, update_old, subdir

| Argument | Description |
|---|---|
| Source_dir | A string whose value is the name of the "master" directory containing files and subdirectories that you want to replicate |
| Target_dir | A string whose value is the name of the target directory to which .files and subdirectories are replicated |
| Add_new | A boolean whose value should be TRUE if you want to replicate files that exist in the source_dir, and FALSE otherwise |
| Delete_missing | A boolean whose value should be TRUE if you want to delete from the target_dir directory these files that target_dir but do not exist in the source_dir, and FALSE otherwise |
| Update_old | A boolean whose value should be TRUE if you want to update older versions of files in the target_dir directory |

| | |
|---|---|
| | update older versions of files in the target_dir directory, and FALSE otherwise. If file is considered as old if it exists in both target_dir and source_dir directories and target_dir version of that files has a time older than the version from the source_dir directory. |
| Subdir | A boolean whose value should be TRUE if you want to update recursive subdirectories. Note that if you enable recursion then all other replication options described above apply to all subdirectories of all nesting levels starting with the Source_dir. |

**Return value:** None.

**Usage:** Use SyncLocalDir statement to automate synchronization and replication for a group of files.
The directory you are copying files from is also known as the **master directory** or **primary site**-replication

24x7 Scheduler logs all file synchronization and replication details to the SYNC.LOG file in the 24x7 Scheduler's installation directory.

**Tip:**

- To update only outdated files set Update_old to TRUE and set both Add_new and Delete_missing to FALSE.
- To perform full 2 way file synchronization between directory <one> and another directory <two>:
  First run SyncLocalDir using <one> as the "master" with Delete_missing set to FALSE and everything else set to TRUE. Repeat SyncLocalDir using <two> as the "master" with Delete_missing set to FALSE and everything else set to TRUE.

**See also:**

SyncFTPDir
SyncRemoteDir
FileCopyEx
FileMoveEx

# SyncRemoteDir

**Description:** Synchronizes and replicates files across two directories residing on local and remote computers provided 24x7 Remote Agent is running at a given remote computer.

**Syntax:** SyncRemoteDir master, agent, source_dir, target_dir, add_new, delete_missing, update_old, subdir

| Argument | Description |
|---|---|
| Master | A string whose value instructs 24x7 Schedule which computer is the "master" computer containing files and subdirectories that you want to replicate.<br>The following values are supported:<br>• "REMOTE" - the remote computer is "master"<br>• "LOCAL" the local computer is "master" |
| Agent | A string whose value is the name of 24x7 Remote Agent or 24x7 Master Scheduler as it is specified in the Remote Agent profile. |

| | |
|---|---|
| Source_dir | A string whose value is the name of the "master" directory containing files and subdirectories that you want to replicate |
| Target_dir | A string whose value is the name of the target directory to which .files and subdirectories are replicated |
| Add_new | A boolean whose value should be TRUE if you want to replicate files that exist in the source_dir, and FALSE otherwise |
| Delete_missing | A boolean whose value should be TRUE if you want to delete from the target_dir directory these files that target_dir but do not exist in the source_dir, and FALSE otherwise |
| Update_old | A boolean whose value should be TRUE if you want to update older versions of files in the target_dir directory, and FALSE otherwise. If file is considered as old if it exists in both target_dir and source_dir directories and target_dir version of that files has a time older than the version from the source_dir directory. |
| Subdir | A boolean whose value should be TRUE if you want to update recursive subdirectories. Note that if you enable recursion then all other replication options described above apply to all subdirectories of all nesting levels starting with the Source_dir. |

**Return value:** None.

**Usage:** Use SyncRemoteDir statement to automate synchronization and replication for a group of files residing on different computers.
The directory you are copying files from is also known as the **master directory** or **primary site**-replication

24x7 Scheduler logs all file synchronization and replication details to the SYNC.LOG file in the 24x7 Scheduler's installation directory.

**Tip:**
- To update only outdated files set Update_old to TRUE and set both Add_new and Delete_missing to FALSE.
- To perform full 2 way file synchronization between directory <one> residing on the remote computer and directory <two> residing on the local computer:
  First run SyncRemoteDir using "REMOTE" for the Master with Delete_missing set to FALSE and everything else set to TRUE. Repeat SyncRemoteDir using "LOCAL" for the Master with Delete_missing set to FALSE and everything else set to TRUE.

**See also:**
SyncFTPDir
SyncLocalDir
FileTransfer

# Dir

**Description:** Returns comma-separated list of files in the specified directory.

**Syntax:** Dir file_mask, return

| Argument | Description |
|---|---|
| File_mask | A string whose value is the DOS file mask that you want to use to list file. File_mask can contain wildcard characters (* and ?). File_mask can contain full or partial file path. |
| Return | A string variable that receives the returned value. |

**Return value:** String. Returns comma-separated list of files.

**Usage:** The Dir statement is equivalent to DOS *dir* command.
To get the list of all files use *.* file mask. If you don't include file path to the file_mask then Dir statement returns files from the DOS current directory.

**See also:**
CD
FileSearchEx
FileFindFirst
FTPDir
RemoteDir

# FTPDir

**Description:** Returns comma-separated list of files in the specified directory on the specified FTP server

**Syntax:** FTPDir server, user, password, file_mask, return

| Argument | Description |
|---|---|
| Server | A string whose value is host name of an FTP server (for example, ftp.microsoft.com) or the IP number of the site in ASCII dotted-decimal format (for example, 11.0.1.45) |
| User | A string whose value is the name of the user to log on |
| Password | A string whose value is the password to use to log on |
| File_mask | A string whose value is the file mask that you want to use to list file. File_mask can contain wildcard characters (* and ?). File_mask can contain full or partial file path.<br>File_mask must be a valid FTP host Operation System file mask. |
| Return | A string variable that receives the returned value. |

**Return value:** String. Returns comma-separated list of files.

**Usage:** On DOS/Windows based FTP hosts the FTPDir statement is equivalent to DOS *dir* command. For most UNIX flavors, the FTPDir statement is equivalent to UNIX *ls* command.
If you don't include file path to the file_mask then FTPDir statement returns files from the FTP server current directory.

**See also:**
FTPFileExists
Dir
RemoteDir

## RemoteDir

**Description:** Returns comma-separated list of files in the specified directory on the specified remote computer running 24x7 Remote Agent

**Syntax:** RemoteDir agent, file_mask, return

| Argument | Description |
|---|---|
| Agent | A string whose value is the name of the Remote Agent as it is specified in the Remote Agent profile. |
| File_mask | A string whose value is the file mask that you want to use to list file. File_mask can contain wildcard characters (* and ?). File_mask can contain full or partial file path. |
| Return | A string variable that receives the returned value. |

**Return value:** String. Returns comma-separated list of files.

**Usage:** The RemoteDir statement is equivalent to DOS *dir* command executed on the remote computer hosting 24x7 Remote Agent specified by Agent.
If you don't include file path to the file_mask then RemoteDir statement returns files from the Agent's computer current directory.

**See also:**
FileTransfer
Dir
FTPDir

# FTP statements

The same statements can be used for both regular FTP and Secure FTP automation. Call the FTPConfig statement in the beginning of a job to specify which protocol to use.

# FTPConfig

**Description:** Set various parameters for the subsequent FTP operations.

**Syntax:** FTPConfig property, new_value

| Argument | Description |
|---|---|
| Property | A string whose value is the name of the property for the FTP session that you want to change. The following properties are supported:<br>• "TRANSFER MODE"<br>• "LIST SEPARATOR"<br>• "PORT"<br>• "CONNECTION TYPE"<br>• "FTP PROTOCOL"<br>• "PRESERVE FILE TIMES"<br>• "SET TIME COMMAND"<br>• "TIME FORMAT"<br>• "TIME OFFSET"<br>• "FTP YEAR BUG" |
| New_value | A string whose value is the new value for the property that you want to change.<br><br>The following values are supported for the "TRANSFER MODE" property:<br>• "ASCII"<br>• "BINARY"<br>The default value is "BINARY". The specified "TRANSFER MODE" is used for all subsequent FTP commands executed in the same job.<br><br>For the "LIST SEPARATOR" property, you can specify any desired symbol that you will use to separate multiple files when performing various multi-file FTP operations. The default value for the "LIST SEPARATOR" is comma. For more information, see **FTPAppendFile, FTPGetFile**, **FTPPutFile**, **FTPREsumeFile** and **FTPDeleteFile** statements. This parameter is used for all subsequent FTP commands executed in the same job.<br><br>Use the "PORT" property, to specify which port your want to use for all subsequent FTP commands executed in the same job. This setting applies to all statements that begin with FTP prefix and to the **SyncFTPDir** and **CompareFTPDir** statements. If you do not change this property, the default FTP port 21 is used for FTP operations. |

| | |
|---|---|
| | The following values are supported for the "CONNECTION TYPE " property:<br>• "PASSIVE"<br>• "ACTIVE"<br>The default value is "ACTIVE". Use this property, to specify whether your want to use active or passive FTP connection mode for all subsequent FTP commands executed in the same job. This setting applies to all statements that begin with FTP prefix and to the **SyncFTPDir** and **CompareFTPDir** statements. If you do not change this property, the default active mode is used for FTP operations.<br><br>The following values are supported for the "FTP PROTOCOL " property:<br>• "SECURE"<br>• "NON SECURE"<br>The default value is "NON SECURE". Use this property, to specify whether your want to use secure or non-secure FTP protocol for all subsequent FTP commands executed in the same job. This setting applies to all statements that begin with FTP prefix and to the **SyncFTPDir** and **CompareFTPDir** statements. If you do not change this property, the default non-secure FTP protocol is used for FTP operations.<br><br>Properties "PRESERVE FILE TIMES", "SET TIME COMMAND", "TIME FORMAT", "TIME OFFSET" are used together as a group of properties that control when and how to set date/time of transferred files. These properties are not used with secure FTP protocol.<br><br>The following values are supported for the "PRESERVE FILE TIMES" property:<br>• "TRUE"<br>• "FALSE"<br>The default value is "FALSE". For more information see the following **Usage** section. Use this property, to specify whether your want to preserve times of uploaded and downloaded files.<br>This property is not used by the secure FTP protocol.<br><br>The following values are supported for the "SET TIME COMMAND" property:<br>• "" (an empty string)<br>• "[user specified host Operation System command]"<br>The default value is "" which instructs the script engine to use the extended version of the MDTM command supported by modern FTP protocols. For more information see the following **Usage** section. |

| | |
|---|---|
| | If you specify some other non-empty value for "SET TIME COMMAND" property the script engine attempts to execute that command as a FTP host operation system command. For this purpose it executes FTP SITE EXEC command following by the specified host command. In order for the host command to be successfully executed your FTP server must support SITE EXEC command and this feature must not be disabled.<br><br>The "TIME FORMAT" property controls time format used with the user-defined command specified in the "SET TIME COMMAND" property. For more information see the following **Usage** section. The default value for "TIME FORMAT" property is YYYYMMDDHHMM.SS.<br><br>The "TIME OFFSET" property can be used to specify the difference in time between the host and the target computer. In other words if the processing computer and the FTP server computer run in different time zones you can use this property to specify the time difference. Generally speaking you can use this property to affect how the file time is set when the value of "PRESERVE FILE TIMES" property is set to TRUE. Specify offset value in seconds. The default value of the "TIME OFFSET" property is 0. You can specify both positive and negative values. A positive value causes the target file time to be adjusted forward; a negative value causes the target file time to be adjusted backward. For more information see the following **Usage** section.<br><br>The "FTP YEAR BUG" parameter can be used to correct incorrect behavior on Windows systems running Internet Explorer 5.0, 5.01, and 5.5. On such systems, **FTPFileDateTime**, **SyncFTPDir** and **CompareFTPDir** statements can incorrectly determine file date information for files that are stored on a UNIX-based File Transfer Protocol (FTP) server (or an FTP server that is running Internet Information Services [IIS] that is running in UNIX folder listing style mode), the year information may appear as one year later or one year earlier than the correct year. This bug is documented in Microsoft Knowledge Base Article – 284455. For more information please visit Microsoft Product Support Services site.<br><br>This bug can cause **SyncFTPDir** to synchronize wrong files. It is recommended that in order to correct the problem you upgrade to Microsoft Internet Explorer 6.0 or later. As a temporary workaround you can use the "FTP YEAR BUG" parameter, which will force FTP statements to automatically adjust file date. The default value for the "FTP YEAR BUG" parameter in 24x7 Scheduler version 3.4.3 and earlier was –1. Starting with version 3.4.4 the default value is 0 meaning that no adjustment is needed. You can set this parameter to the following values<br>• "-1"<br>• "1"<br>• "0"<br><br>If a non-zero value is set for this parameter, FTP statements will automatically adjust year portion of file |

| | |
|---|---|
| | dates by the specified value. |

**Return value:** None.

**Usage:** Use FTPConfig statement with the "TRANSFER MODE" property to configure transfer mode when downloading or uploading files using FTPGetFile and FTPPutFile statements. Both parameter name and new value are case insensitive. Use FTPConfig statement with the "LIST SEPARATOR" property to configure multi-file FTP operations.

"TRANSFER MODE" setting determines which transfer mode will be set when you initiate a file transfer. It can be one of the following:

**ASCII**: This mode is used for transferring text files between sites running different operating systems. It will take care of all the necessary translation to make text readable. However, use it with caution since binary files will be corrupted if transferred in this mode. Text files with non-English characters in it will also be corrupted.

**Binary**: This is the most frequently used transfer mode. It transfers raw data without any translation.

"LIST SEPARATOR" setting overrides the default comma symbol used to separate names of files in other FTP statements. Specify some other symbol if a file you need to process contains comma as a part of the file name.

"PORT" setting overrides the default FTP port number.

"CONNECTION TYPE" setting overrides the default FTP connection mode.

"FTP PROTOCOL" setting overrides the default FTP protocol used for FTP operations. For more information on secure FTP protocols read RFC-2228 specification.

**Important Note:**

Secure FTP protocol does not currently support ASCII transfer mode, that's why all file transfers are always performed using Binary transfer mode regardless of the "TRANSFER MODE" setting.

Properties "PRESERVE FILE TIMES", "SET TIME COMMAND", "TIME FORMAT", "TIME OFFSET" are used together as a group of properties that control when and how to set date/time of transferred files. These properties are not used with secure FTP protocol.

Use these properties, to specify whether your want to preserve times of uploaded and downloaded files. By default all transferred files receive system date/time on the destination computer at the time of the transfer. When the "PRESERVE FILE TIMES" property is set to "TRUE", job engine automatically executes additional commands required to change the times of transferred files to match times of the original files.

**Important note:**

Not all FTP servers support methods for changing times of uploaded files. See description of the "SET TIME COMMAND" property for more information about supported methods.

If an empty string is specified for "SET TIME COMMAND" property the script engine attempts to use the extended version of the FTP MDTM command supported by modern FTP protocols. Please note that the FTP MDTM command is not understood by all FTP servers. The command is executed internally as

*MDTM [TIME] [FILE]*

In job run-time the [TIME] parameter is substituted with the time of the source file in YYYYMMDDhhmmss format and [FILE] is substituted with the name of the target file.

For example if file time is 16-June-2001 2:00:05 PM, time format is YYYYMMDDHHMM.SS and file name is test.txt the final command text will be sent to FTP server as

```
MDTM 20010616140005 test.txt
```

If you specify some other non-empty value for "SET TIME COMMAND" property the script engine attempts to execute that command as a FTP host operation system command. For this purpose it executes FTP SITE EXEC command following by the specified host command. In order for the host command to be successfully executed your FTP server must support SITE EXEC command and this feature must not be disabled. The user specified operation system command may include [TIME] and [FILE] macros that will be automatically replaced in the run-time with the source file time and target file name. The format of the file time value is driven by the "TIME FORMAT" property.

For example many UNIX systems support the touch command that can be used to change file times.

Example of the complete command:

```
touch -mct [TIME] [FILE]
```

This example command will be executed as

```
SITE EXEC touch -mct [TIME] [FILE]
```

For example if file time is 16-June-2001 2:00:05 PM, time format is YYYYMMDDHHMM.SS and file name is test.txt the final command text will be sent to the FTP server as

```
SITE EXEC touch –mct 200106161400.05 test.txt
```

Use the "TIME FORMAT" property to specify time format to be used with the user-defined command specified in the "SET TIME COMMAND" property. See previous 4 paragraphs for more information.

**Important note:**

The default value for the "FTP YEAR BUG" parameter in 24x7 Scheduler version 3.4.3 and earlier was –1. Starting with version 3.4.4 the default value is 0 meaning that no adjustment is needed.

**See also:**

FTPGetFile
FTPPutFile
FTPResumeFile
FTPAppendFile
SyncFTPDir

# FTPDir

**Description:** Returns comma-separated list of files in the specified directory on the specified FTP server

**Syntax:** FTPDir server, user, password, file_mask, return

| Argument | Description |
|---|---|
| Server | A string whose value is host name of an FTP server (for example, ftp.microsoft.com) or the IP number of the site in ASCII dotted-decimal format (for example, 11.0.1.45) |
| User | A string whose value is the name of the user to log on |
| Password | A string whose value is the password to use to log on |
| File_mask | A string whose value is the file mask that you want to use |

| | |
|---|---|
| | to list file. File_mask can contain wildcard characters (* and ?). File_mask can contain full or partial file path.<br><br>File_mask must be a valid FTP host Operation System file mask. |
| Return | A string variable that receives the returned value. |

**Return value:** String. Returns comma-separated list of files.

**Usage:** On DOS/Windows based FTP hosts the FTPDir statement is equivalent to DOS *dir* command. For most UNIX flavors, the FTPDir statement is equivalent to UNIX *ls* command.
If you don't include file path to the file_mask then FTPDir statement returns files from the FTP server current directory.

By default 24x7 Scheduler uses standard non-secure FTP protocol for all FTP operations. If you are working with a secure FTP server (SFTP server) use **FTPConfig** statement to switch FTP protocol.

By default 24x7 Scheduler performs all FTP operations using default port for FTP servers. To specify a different port number use **FTPConfig** statement.

**Important Notes:**

- Read **File Caching Internet Options** topic if you experience various "cannot create file/directory because it already exists" errors or files do not get updated during FTP downloads.

**See also:**

FTPFileExists
Dir
RemoteDir

# FTPDirCreate

**Description:** Creates a new directory on remote FTP server.

**Syntax:** FTPDirCreate server, user, password, dir

| Argument | Description |
|---|---|
| server | A string whose value is host name of an FTP server (for example, ftp.microsoft.com) or the IP number of the site in ASCII dotted-decimal format (for example, 11.0.1.45) |
| User | A string whose value is the name of the user to log on |
| password | A string whose value is the password to use to log on |
| dir | A string whose value is the full name of the directory to be created |

**Return value:** None

**Usage:** By default 24x7 Scheduler uses standard non-secure FTP protocol for all FTP operations. If you are working with a secure FTP server (SFTP server) use **FTPConfig** statement to switch FTP protocol.

By default 24x7 Scheduler performs all FTP operations using default port for FTP servers. To specify a different port number use **FTPConfig** statement.

**Important Notes:**

- Directory and file names are case-sensitive on all FTP server platforms.
- Read **File Caching Internet Options** topic if you experience various "cannot create file/directory because it already exists" errors or files do not get updated during FTP downloads.

**See also:**

FTPDirDelete
FTPDir

# FTPDirDelete

**Description:** Delete an existing directory on remote FTP server. If that directory contains files or subdirectories, 24x7 Scheduler deletes them recursively.

**Syntax:** FTPDirDelete server, user, password, dir

| Argument | Description |
|---|---|
| server | A string whose value is host name of an FTP server (for example, ftp.microsoft.com) or the IP number of the site in ASCII dotted-decimal format (for example, 11.0.1.45) |
| user | A string whose value is the name of the user to log on |
| password | A string whose value is the password to use to log on |
| dir | A string whose value is the full name of the directory to be deleted |

**Return value:** None

**Usage:** By default 24x7 Scheduler uses standard non-secure FTP protocol for all FTP operations. If you are working with a secure FTP server (SFTP server) use **FTPConfig** statement to switch FTP protocol.

By default 24x7 Scheduler performs all FTP operations using default port for FTP servers. To specify a different port number use **FTPConfig** statement.

By default 24x7 Scheduler uses standard non-secure FTP protocol for all FTP operations. If you are working with a secure FTP server (SFTP server) use **FTPConfig** statement to switch FTP protocol.

By default 24x7 Scheduler performs all FTP operations using default port for FTP servers. To specify a different port number use **FTPConfig** statement.

**Important Notes:**

- Directory and file names are case-sensitive on all FTP server platforms.
- Read **File Caching Internet Options** topic if you experience various "cannot create file/directory because it already exists" errors or files do not get updated during FTP downloads.

**See also:**

FTPDirCreate

FTPDeleteFile
FTPDir

# FTPDeleteFile

**Description:** Deletes the specified file on the specified remote FTP server.

**Syntax:** FTPDeleteFile server, user, password, file

| Argument | Description |
|---|---|
| Server | A string whose value is host name of an FTP server (for example, ftp.microsoft.com) or the IP number of the site in ASCII dotted-decimal format (for example, 11.0.1.45) |
| User | A string whose value is the name of the user to log on |
| password | A string whose value is the password to use to log on |
| file | A string whose value is the name of the file that you want to delete |

**Return value:** None.

**Usage:** file can be either partially or fully qualified file name relative to the current directory. A backslash (\) or forward slash (/) can be used as the directory separator for the name. The FTPDeleteFile statement translates the directory name separators to the appropriate character before they are used.

By default 24x7 Scheduler uses standard non-secure FTP protocol for all FTP operations. If you are working with a secure FTP server (SFTP server) use **FTPConfig** statement to switch FTP protocol.

By default 24x7 Scheduler performs all FTP operations using default port for FTP servers. To specify a different port number use **FTPConfig** statement.

 **Tip:**
FTPDeleteFile statement can delete multiple files in one pass. This is more efficient than calling FTPDeleteFile for each file separately, which requires a separate FTP connection for every file. To delete multiple files in one pass, specify multiple files names in the file parameter as a comma separated list.

 **Important Notes:**

- Read **File Caching Internet Options** topic if you experience various "cannot create file/directory because it already exists" errors or files do not get updated during FTP downloads.

**See also:**

FTPGetFile

# FTPFileDateTime

**Description:** Reports date and time of the specified file on the specified FTP server.

**Syntax:** FTPFileDateTime server, user, password, file, return

| Argument | Description |
|---|---|
| server | A string whose value is host name of an FTP server (for example, ftp.microsoft.com) or the IP number of the site in ASCII dotted-decimal format (for example, 11.0.1.45) |
| user | A string whose value is the name of the user to log on |
| password | A string whose value is the password to use to log on |
| file | A string whose value is the name of the file that you want to check |
| return | A datetime variable that receives the returned value |

**Return value:** Datetime. Returns date/time of the specified file.

**Usage:** file can be either partially or fully qualified file name relative to the current directory. A backslash (\) or forward slash (/) can be used as the directory separator for the name. The FTPFileDateTime statement translates the directory name separators to the appropriate character before they are used.

By default 24x7 Scheduler uses standard non-secure FTP protocol for all FTP operations. If you are working with a secure FTP server (SFTP server) use **FTPConfig** statement to switch FTP protocol.

By default 24x7 Scheduler performs all FTP operations using default port for FTP servers. To specify a different port number use **FTPConfig** statement.

 **Important Notes:**

- File names are case-sensitive on all FTP server platforms.
- Read **File Caching Internet Options** topic if you experience various "cannot create file/directory because it already exists" errors or files do not get updated during FTP downloads.

**See also:**

FTPFileSize

FTPFileExists

# FTPFileSize

**Description:** Reports size of the specified file on the specified FTP server.

**Syntax:** FTPFileSize server, user, password, file, return

| Argument | Description |
|---|---|
| server | A string whose value is host name of an FTP server (for example, ftp.microsoft.com) or the IP number of the site in ASCII dotted-decimal format (for example, 11.0.1.45) |
| user | A string whose value is the name of the user to log on |
| password | A string whose value is the password to use to log on |
| file | A string whose value is the name of the file that you want to check |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns size of the specified file.

**Usage:** file can be either partially or fully qualified file name relative to the current directory. A backslash (\) or forward slash (/) can be used as the directory separator for the name. The FTPFileSize statement translates the directory name separators to the appropriate character before they are used.

By default 24x7 Scheduler uses standard non-secure FTP protocol for all FTP operations. If you are working with a secure FTP server (SFTP server) use **FTPConfig** statement to switch FTP protocol.

By default 24x7 Scheduler performs all FTP operations using default port for FTP servers. To specify a different port number use **FTPConfig** statement.

File names are case-sensitive on all FTP server platforms.

**See also:**

FTPFileDateTime

FTPFileExists

# FTPFileExists

**Description:** Reports whether the specified file exists on the specified remote FTP server.

**Syntax:** FTPFileExists server, user, password, file, return

| Argument | Description |
|---|---|
| server | A string whose value is host name of an FTP server (for example, ftp.microsoft.com) or the IP number of the site in ASCII dotted-decimal format (for example, 11.0.1.45) |
| user | A string whose value is the name of the user to log on |
| password | A string whose value is the password to use to log on |
| file | A string whose value is the name of the file that you want to check |
| return | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE if the file exists, FALSE if it does not exist.

**Usage:** file can be either partially or fully qualified file name relative to the current directory. A backslash (\) or forward slash (/) can be used as the directory separator for the name. The FTPFileExists statement translates the directory name separators to the appropriate character before they are used.

By default 24x7 Scheduler uses standard non-secure FTP protocol for all FTP operations. If you are working with a secure FTP server (SFTP server) use **FTPConfig** statement to switch FTP protocol.

By default 24x7 Scheduler performs all FTP operations using default port for FTP servers. To specify a different port number use **FTPConfig** statement.

 **Important Notes:**

- File names are case-sensitive on all FTP server platforms.
- Read **File Caching Internet Options** topic if you experience various "cannot create file/directory because it already exists" errors or files do not get updated during FTP downloads.

**See also:**

FTPGetFile

FTPDir

# FTPGetFile

**Description:** Retrieves a file from the specified FTP server and stores it under the specified file name, creating a new local file in the process.

**Syntax:** FTPGetFile server, user, password, source, target

| Argument | Description |
|---|---|
| server | A string whose value is host name of an FTP server (for example, ftp.microsoft.com) or the IP number of the site in ASCII dotted-decimal format (for example, 11.0.1.45) |
| user | A string whose value is the name of the user to log on |
| password | A string whose value is the password to use to log on |
| source | A string whose value is the name of the file to retrieve from the remote system |
| target | A string whose value is the name of the file to create on the local system |

**Return value:** None.

**Usage:** Both source and target file can be either partially or fully qualified file names relative to the current directory. A backslash (\) or forward slash (/) can be used as the directory separator for either name. The FTPGetFile statement translates the directory name separators to the appropriate character before they are used. By default FTPGetFile statement uses binary transfer mode. To change transfer mode to ASCII, use **FTPConfig** statement to modify the "TRANSFER MODE" setting.

By default 24x7 Scheduler uses standard non-secure FTP protocol for all FTP operations. If you are working with a secure FTP server (SFTP server) use **FTPConfig** statement to switch FTP protocol.

By default 24x7 Scheduler performs all FTP operations using default port for FTP servers. To specify a different port number use **FTPConfig** statement.

File names are case-sensitive on all FTP server platforms.

**Tip:**
FTPGetFile statement can transfer multiple files in one pass. This is more efficient than calling FTPGetFile for each file separately, which requires a separate FTP connection for every file. To transfer multiple files in one pass, specify the source files as a comma separated list. The target files must be also specified as a comma separated list. Make sure to specify the same number of file names in the source and target file lists.

**Important Notes:**

- Secure FTP protocol does not currently support ASCII transfer mode, that's why all file transfers are always performed using Binary transfer mode regardless of the "TRANSFER MODE" setting.
- Read **File Caching Internet Options** topic if you experience various "cannot create file/directory because it already exists" errors or files do not get updated during FTP downloads.

**See also:**

FTPPutFile
FTPConfig
SyncFTPDir

# FTPResumeFile

**Description:** Retrieves a file from the specified FTP server and stores it under the specified file name, creating a new local file in the process or appending to local file if it already exists.

**Syntax:** FTPResumeFile server, user, password, source, target

| Argument | Description |
|---|---|
| server | A string whose value is host name of an FTP server (for example, ftp.microsoft.com) or the IP number of the site in ASCII dotted-decimal format (for example, 11.0.1.45) |
| user | A string whose value is the name of the user to log on |
| password | A string whose value is the password to use to log on |
| source | A string whose value is the name of the file to retrieve from the remote system |
| target | A string whose value is the name of the file to create on the local system |

**Return value:** None.

**Usage:** Both source and target file can be either partially or fully qualified file names relative to the current directory. A backslash (\) or forward slash (/) can be used as the directory separator for either name. The FTPResumeFile

statement translates the directory name separators to the appropriate character before they are used. By default FTPResumeFile statement uses binary transfer mode. To change transfer mode to ASCII, use **FTPConfig** statement to modify the "TRANSFER MODE" setting.

FTPResumeFile statement is identical to **FTPGetFile** statement except that it attempts to resume broken downloads or perform incremental downloads of files whose size have increased since the last download.

By default 24x7 Scheduler performs all FTP operations using default port for FTP servers. To specify a different port number use **FTPConfig** statement.

File names are case-sensitive on all FTP server platforms.

 **Tip:**
FTPResumeFile statement can transfer multiple files in one pass. This is more efficient than calling FTPResumeFile for each file separately, which requires a separate FTP connection for every file. To transfer multiple files in one pass, specify the source files as a comma separated list. The target files must be also specified as a comma separated list. Make sure to specify the same number of file names in the source and target file lists.

 **Important Note:**

- Not all FTP servers support RESUME operations. An error will occur if you attempt to run FTPResumeFile statement for a server that does not support this feature.
- FTPResumeFile statement is supported only in standard FTP mode. It is not supported in Secure FTP mode (SFTP), because Secure FTP protocol does not currently provide support for RESUME operations.
- Read **File Caching Internet Options** topic if you experience various "cannot create file/directory because it already exists" errors or files do not get updated during FTP downloads.
- You must have Microsoft ® Internet Explorer version 5.0 or better installed on the computer in order to use FTPCommand, FTPResumeFile and FTPAppendFile statements.

**See also:**

FTPGetFile
FTPPutFile
FTPConfig
SyncFTPDir

# FTPPutFile

**Description:** Transfers a file from the local system to specified remote FTP server and stores it under the specified file name, creating a new remote file in the process.

**Syntax:** FTPPutFile server, user, password, source, target

| Argument | Description |
|---|---|
| server | A string whose value is host name of an FTP server (for example, ftp.microsoft.com) or the IP number of the site in ASCII dotted-decimal format (for example, 11.0.1.45) |
| user | A string whose value is the name of the user to log on |

| | |
|---|---|
| password | A string whose value is the password to use to log on |
| source | A string whose value is the name of the file to transfer from the local system |
| target | A string whose value is the name of the file to create on the remote system |

**Return value:** None.

**Usage:** Both source and target file can be either partially or fully qualified file names relative to the current directory. A backslash (\) or forward slash (/) can be used as the directory separator for either name. The FTPPutFile statement translates the directory name separators to the appropriate character before they are used. By default FTPPutFile statement uses binary transfer mode. To change transfer mode to ASCII, use **FTPConfig** statement to modify the "TRANSFER MODE" setting.

By default 24x7 Scheduler uses standard non-secure FTP protocol for all FTP operations. If you are working with a secure FTP server (SFTP server) use **FTPConfig** statement to switch FTP protocol.

By default 24x7 Scheduler performs all FTP operations using default port for FTP servers. To specify a different port number use **FTPConfig** statement.

File names are case-sensitive on all FTP server platforms.

 **Tip:**
FTPPutFile statement can transfer multiple files in one pass. This is more efficient than calling FTPPutFile for each file separately, which requires a separate FTP connection for every file. To transfer multiple files in one pass, specify the source files as a comma separated list. The target files must be also specified as a comma separated list. Make sure to specify the same number of file names in the source and target file lists.

 **Important Notes:**

- Secure FTP protocol does not currently support ASCII transfer mode, that's why all file transfers are always performed using Binary transfer mode regardless of the "TRANSFER MODE" setting.
- Read **File Caching Internet Options** topic if you experience various "cannot create file/directory because it already exists" errors or files do not get updated during FTP downloads.

**See also:**

FTPGetFile
FTPConfig
SyncFTPDir

# FTPAppendFile

**Description:** Transfers a file from the local system to specified remote FTP server and stores it under the specified file name, creating a new remote file in the process or appending data to an existing remote file.

**Syntax:** FTPAppendFile server, user, password, source, target

| Argument | Description |
|---|---|
| server | A string whose value is host name of an FTP server (for example, ftp.microsoft.com) or the IP number of the site in ASCII dotted-decimal format (for example, 11.0.1.45) |
| user | A string whose value is the name of the user to log on |
| password | A string whose value is the password to use to log on |
| source | A string whose value is the name of the file to transfer from the local system |
| target | A string whose value is the name of the file to create on the remote system |

**Return value:** None.

**Usage:** Both source and target file can be either partially or fully qualified file names relative to the current directory. A backslash (\) or forward slash (/) can be used as the directory separator for either name. The FTPAppendFile statement translates the directory name separators to the appropriate character before they are used. By default FTPappendFile statement uses binary transfer mode. To change transfer mode to ASCII, use **FTPConfig** statement to modify the "TRANSFER MODE" setting.

FTPAppendFile statement is identical to **FTPPutFile** statement except that it attempts to append to an existing file on the FTP server rather than overwrite it.

By default 24x7 Scheduler performs all FTP operations using default port for FTP servers. To specify a different port number use **FTPConfig** statement.

File names are case-sensitive on all FTP server platforms.

**Tip:**
FTPAppendFile statement can transfer multiple files in one pass. This is more efficient than calling FTPAppendFile for each file separately, which requires a separate FTP connection for every file. To transfer multiple files in one pass, specify the source files as a comma separated list. The target files must be also specified as a comma separated list. Make sure to specify the same number of file names in the source and target file lists.

**Important Note:**

- Not all FTP servers support APPEND operations. An error will occur if you attempt to run FTPAppendFile statement for a server that does not support this feature.
- FTPAppendFile statement is supported only in standard FTP mode. It is not supported in Secure FTP mode (SFTP), because Secure FTP protocol does not currently provide support for APPEND operations.
- Read **File Caching Internet Options** topic if you experience various "cannot create file/directory because it already exists" errors or files do not get updated during FTP downloads.
- You must have Microsoft ® Internet Explorer version 5.0 or better installed on the computer in order to use FTPCommand, FTPResumeFile and FTPAppendFile statements.

**See also:**

FTPPutFile
FTPGetFile
FTPConfig
SyncFTPDir

# FTPRenameFile

**Description:** Renames the specified remote file on the specified FTP server.

**Syntax:** FTPRenameFile server, user, password, oldname, newname

| Argument | Description |
|---|---|
| Server | A string whose value is host name of an FTP server (for example, ftp.microsoft.com) or the IP number of the site in ASCII dotted-decimal format (for example, 11.0.1.45) |
| User | A string whose value is the name of the user to log on |
| Password | A string whose value is the password to use to log on |
| Oldname | A string whose value is the name of the file to rename |
| Newname | A string whose value is the new name of the file |

**Return value:** None.

**Usage:** Both oldname and newname file can be either partially or fully qualified file names relative to the current directory. A backslash (\) or forward slash (/) can be used as the directory separator for either name. The FTPRenameFile statement translates the directory name separators to the appropriate character before they are used.

By default 24x7 Scheduler uses standard non-secure FTP protocol for all FTP operations. If you are working with a secure FTP server (SFTP server) use **FTPConfig** statement to switch FTP protocol.

By default 24x7 Scheduler performs all FTP operations using default port for FTP servers. To specify a different port number use **FTPConfig** statement.

File names are case-sensitive on all FTP server platforms.

**See also:**

FTPPutFile
FTPGetFile
FTPDeleteFile

# FTPCommand

**Description:** Executes arbitrary commands directly on the specified FTP server.

**Syntax:** FTPCommand server, user, password, command

| Argument | Description |
|---|---|
| server | A string whose value is host name of an FTP server (for example, ftp.microsoft.com) or the IP number of the site in ASCII dotted-decimal format (for example, 11.0.1.45) |
| user | A string whose value is the name of the user to log on |
| password | A string whose value is the password to use to log on |
| command | A string whose value is the command that you want to execute on the server |

**Return value:** None

**Usage:** The FTPCommand statement allows you to enter commands directly to the FTP server. The available commands vary depending on the type of server, and can usually be determined by logging on to the server with the command line FTP client and using the "remotehelp" command. A typical output of "remotehelp" command looks like the following:

```
ftp> remotehelp
  The following commands are recognized (* =>'s unimplemented).

   USER    PORT    STOR     MSAM*   RNTO    NLST     MKD     CDUP
   PASS    PASV    APPE     MRSQ*   ABOR    SITE     XMKD    XCUP
   ACCT*   TYPE    MLFL*    MRCP*   DELE    SYST     RMD     STOU
   SMNT*   STRU    MAIL*    ALLO    CWD     STAT     XRMD    SIZE
   REIN*   MODE    MSND*    REST    XCWD    HELP     PWD     MDTM
   QUIT    RETR    MSOM*    RNFR    LIST    NOOP     XPWD
```

By default 24x7 Scheduler performs all FTP operations using default port for FTP servers. To specify a different port number use **FTPConfig** statement.

**Important Notes:**

- Read **File Caching Internet Options** topic if you experience various "cannot create file/directory because it already exists" errors or files do not get updated during FTP downloads.
- Not all FTP servers support QUOTE operation which is used to execute remote commands. An error will occur if you attempt to run FTPCommand statement for a server that does not support this feature.
- FTPCommand statement is supported only in standard FTP mode. It is not supported in Secure FTP mode (SFTP), because Secure FTP protocol does not currently provide support for QUOTE operations.
- The SITE command can be used to execute operation system commands on the remote FTP server computer.

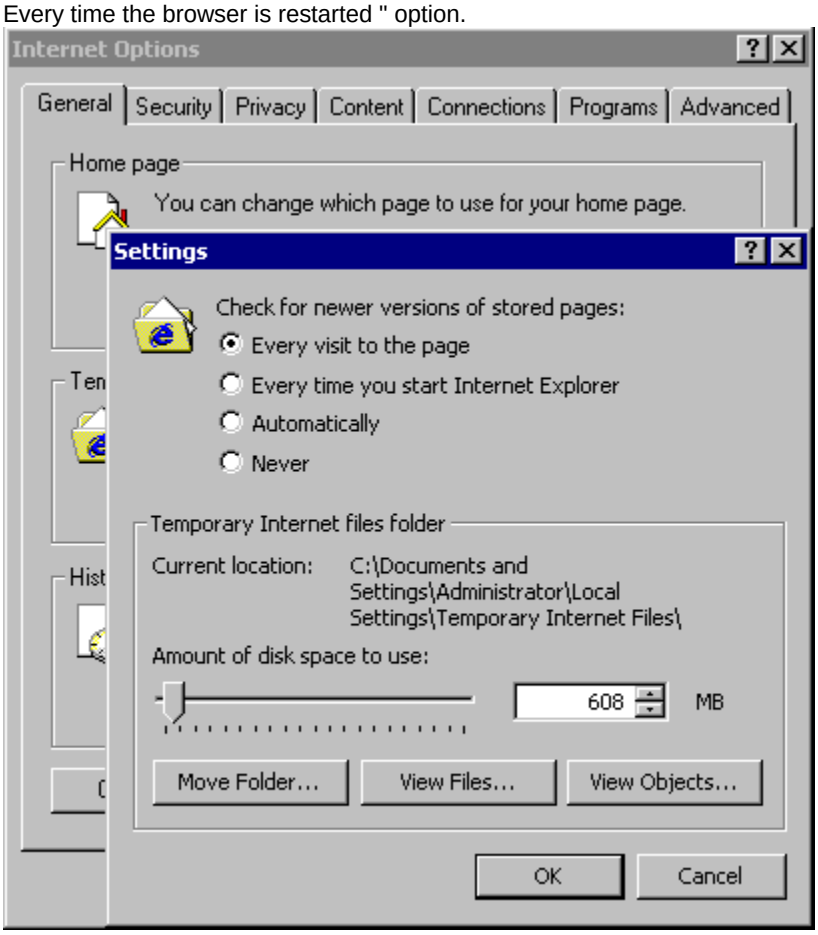
  For example to set read permissions for file *myfile.txt* you can execute the following command (this example is given for a UNIX based FTP server):

  ```
  FTPCommand "myserver", "user", "password", "SITE chmod +r /home/myfile.txt"
  ```

- Executing FTPCommand statement has no effect on subsequent FTP operations because each FTP operation is executed in a separate session.
- You must have Microsoft ® Internet Explorer version 5.0 or better installed on the computer in order to use FTPCommand, FTPResumeFile and FTPAppendFile statements.

# File Caching Internet Options

24x7 Scheduler utilizes a number of Windows API functions for FTP and Internet file operations. As a result of that certain Windows settings can affect job processing within 24x7. If you experience various "cannot create file/directory because it already exists" errors or files do not get updated during FTP downloads the problem may be with your Windows settings (local file cache). In the Windows Control Panel select " Internet Options " applet and then under " Temporary Internet Files " select " settings " and pick the "Every Visit to the page " option rather than the default " Every time the browser is restarted " option.




The problem may be also with your Internet Service Provider (ISP). If your ISP uses some caching techniques, you may be working with your old files for certain period of time (regardless of whether you have updated the files using one of the supported FTP methods or not. But this problem should not exist for more than a few hours. If you are seeing the old file for more than one day, then the problem may not be related to the ISP. If you believe your problem is ISP related contact your ISP support and request to turn off the caching options.

# Job management statements

## JobCreate

**Description:** Creates a new job.

**Syntax:** JobCreate return

| Argument | Description |
|---|---|
| Return | Number |

**Return value:** Number. Returns job ID for the new job. The new job inherites default job properties.

**Usage:** Use JobCreate in your scripts to programmatically create new jobs. Use JobModify statement to setup properties of the created job . Use JobEnable statement with the TRUE status to enable the job after you done with setting job properties.

**Important note:**

- The JobCreate statement is not supported in remote jobs executed on 24x7 Remote Agents. An error “Job not found” will occur if you use this statement. However, it can be used if the remote job is executed on the 24x7 Master Scheduler.

**See also:**

JobDelete
JobEnable
JobModify

# JobDelete

**Description:** Deletes the specified job.

**Syntax:** JobDelete job

| Argument | Description |
|---|---|
| Job | A string whose value is ether job ID or job name |

**Return value:** None

**Usage:** Use JobDelete in your scripts to programmatically delete jobs. JobDelete permanently deletes the specified job from the active job pool. The deleted job cannot be recovered at a later time. Use JobEnable statement with the FALSE state to temporary disable jobs. If the specified job is not found, an error occurs.

**Important note:**

- The JobDelete statement is not supported in remote jobs executed on 24x7 Remote Agents. An error “Job not found” will occur if you use this statement. However, it can be used if the remote job is executed on the 24x7 Master Scheduler.

**See also:**

JobEnable
JobGetStatus

# JobDescribe

**Description:** Describes properties of the specified job.

**Syntax:** JobDescribe job, property, value

| Argument | Description |
|---|---|
| job | A string whose value is ether job ID or job name |
| property | A string whose value is the property that you want to modify. See Job Properties topic for list and description of available properties |
| value | A string variable that receives the returned value. |

**Return value:** A string whose value matches the value for the property. The returned value is always converted to a string.

**Usage:** Use this statement to retrieve job definitions, job schedules and triggers programmatically. For example, you can use JobDescribe to create customizable job monitors.

**Notes:**

- The JobDescribe statement is not supported in remote jobs executed on 24x7 Remote Agents. An error “Job not found” will occur if you use this statement. However, it can be used if the remote job is executed on the 24x7 Master Scheduler.

**See also:**

JobModify

JobEnable

Job Properties

# JobList

**Description:** Returns comma-separated list of job IDs in the active job database.

**Syntax:** JobList folder, return

| Argument | Description |
|---|---|
| folder | A string whose value is the name of the folder in which you want to list jobs. Specify an empty string ("") for the folder to obtain a list of all job IDs in the job database. |
| return | A string variable that receives the returned value. |

**Return value:** A string containing comma-separated list of job IDs.

**Tip:**
- You can use the GetToken and LoopWhile statements to parse the returned job list.

**Notes:**
- The JobList statement is not supported in remote jobs executed on 24x7 Remote Agents. However, it can be used in remote jobs executed on 24x7 Master Schedulers.

**See also:**

JobModify
JobEnable
JobDescribe
Job Properties

# JobEnable

**Description:** Enables or disables the specified job.

**Syntax:** JobEnable job, state

| Argument | Description |
|---|---|
| job | A string whose value is ether job ID or job name |
| state | A boolean whose value is the new job state. Specify TRUE to enable the job or FALSE to disable the job |

**Return value:** None

**Usage:** Use this statement to enable/disable jobs in the active job pool. If the specified job is not found, an error occurs.

**Notes:**
- Jobs disabled by using JobEnable statement remain in the active job pool until job database “save" is invoked via 24x7 GUI or via external interface. On “save" all disabled jobs are purged from the active job pool. Jobs disabled via 24x7 GUI are purged from the active job pool right away. However, disabled jobs are not removed from the job database. To physically remove a job from the job database use JobDelete statement, 24x7 GUI or one of the supported external interfaces.
- The JobEnable statement is not supported in remote jobs executed on 24x7 Remote Agents. An error “Job not found” will occur if you use this statement. However, it can be used if the remote job is executed on the 24x7 Master Scheduler.

**See also:**

JobRun
JobModify

# JobGetStatus

**Description:** Reports current status of the specified job.

**Syntax:** JobGetStatus job, return

| Argument | Description |
|---|---|
| job | A string whose value is ether job ID or job name |
| return | A numeric whose value is the job status. |

**Return value:** Number. Returned job status can be one of the following:

| | |
|---|---|
| -1 | Error job. Job may have this status as a result of incomplete job definition or if a run-time error occurred during last job run. |
| -2 | Job is executing job notification action such as sending email message, executing database command, etc. |
| -3 | Job is running. |
| -5 | First job run is pending. This job hasn't run since the last scheduler restart. |
| -6 | Unknown job status. Status of asynchronous job of “Program” type might be unknown after job start. This status can be also reported in case multiple job instances found in job queues. |
| 0 | Job successfully completed. Note: an asynchronous job of “Program” type also may have this status after it successfully started but before the program actually finished running. That's the nature of asynchronous jobs. |
| Any other number | Successfully completed (applicable to asynchronous jobs of “Program” type only). A job may have this status after it successfully started and is still running. |

**Usage:** Use this statement to retrieve job status programmatically from the script. This statement can be useful in building various job monitors.

**Notes:**

- Use JobDescribe statement to found out job Enabled/Disabled state.
- You cannot change job status directly. Job status is the internal job state maintained by the 24x7 Scheduler.
- The JobGetStatus statement is not supported in remote jobs executed on 24x7 Remote Agents. An error “Job not found” will occur if you use this statement. However, it can be used if the remote job is executed on the 24x7 Master Scheduler.
- An asynchronous job of “Program” type may have 0 status even if the started “program” (process) is still running in any of the following cases: started program spawned one or more child processes after which the main entry point process has terminated; started program is not responding or has security attributes that do not allow querying status of the process.

**See also:**

JobRun
JobDescribe

# JobModify

**Description:** Modifies properties of the specified job.

**Syntax:** JobModify job, property, new_value

| Argument | Description |
|---|---|
| job | A string whose value is ether job ID or job name |
| property | A string whose value is the property that you want to modify. See Job Properties topic for list and description of available properties |
| new_value | A string whose value is the new value for the property |

**Return value:** None

**Usage:** Use this statement to modify job definitions, job schedules and triggers programmatically. For example, you can use JobModify to reschedule jobs for a later run when the exact start time is unknown in design-time.

 **Notes:**

- Modified jobs are updated immediately in both places: the active job pool and the job database. 24x7 Scheduler internally triggers “save" command after every change.
- Side effect: On “save,"all disabled jobs are purged from the active job pool. However, disabled jobs are not removed from the job database. To physically remove a job from the job database use JobDelete statement, 24x7 GUI or one of supported external interfaces.
- The JobModify statement is not supported in remote jobs executed on 24x7 Remote Agents. An error “Job not found” will occur if you use this statement. However, it can be used if the remote job is executed on the 24x7 Master Scheduler.

**See also:**

JobModify
JobDescribe
JobEnable
Job Properties

# JobHold

**Description:** Places the specified job on hold.

**Syntax:** JobHold job_runtime_id

| Argument | Description |
|---|---|
| job_runtime_id | A number whose value is job runtime ID in job queue. Job runtime ID uniquely identifies job instance in job queues. |

**Return value:** None

**Usage:** Use JobHold in your scripts to programmatically delay queued jobs. Job run-time ID is returned by JobSendToQueue statement and can be also obtained using QueueJobList statement. JobHold can delay queued jobs only. A held job cannot start until it is released. the queue manager ignores held jobs. An error occurs if the specified job_runtime_id cannot be found

**Important note:**

- This statement is not supported in remote jobs executed on 24x7 Remote Agents. An error “Job not found” will occur if you use this statement.
- Do not confuse JobHold and JobDisable statements. The last one can be used to disable job definition and remove if from the active job pool.
- This statement is not supported in standalone JAL scripts executed from command line.

**See also:**

QueueJobList
JobSendToQueue
JobEnable
JobRelease
JobRun

# JobRelease

**Description:** Releases the specified job from held state and allows the queue to process this job.

**Syntax:** JobRelease job_runtime_id

| Argument | Description |
|---|---|
| job_runtime_id | A number whose value is job runtime ID in job queue. Job runtime ID uniquely identifies job instance in job queues. |

**Return value:** None

**Usage:** Use JobRelease in your scripts to programmatically releases held jobs. Job run-time ID is returned by JobSendToQueue statement and can be also obtained using QueueJobList statement. When a job is released it goes to the end of the queue. If the queue manager is not busy, it picks the released job and starts running it immediately. If the queue manager is busy running other jobs, the released job will remain in the queue until the queue manager is free to run it. An error occurs if the specified job_runtime_id cannot be found

**Important note:**

- This statement is not supported in remote jobs executed on 24x7 Remote Agents. An error “Job not found” will occur if you use this statement.
- Do not confuse JobRelease and JobEnable statements. The last one can be used to enable previous disabled job definition and place them back into active job pool.
- This statement is not supported in standalone JAL scripts executed from command line.

**See also:**

QueueJobList
JobSendToQueue
JobEnable
JobHold
JobRun

# JobKill

**Description:** Kills specified job if it is already running and removes it from the job queue.

**Syntax:** JobKill job_runtime_id

| Argument | Description |
|---|---|
| job_runtime_id | A number whose value is job runtime ID in job queue. Job runtime ID uniquely identifies job instance in job queues. |

**Return value:** None

**Usage:** Use JobKill in your scripts to programmatically terminate running jobs and/or delete jobs instances from job queues. Job run-time ID is returned by JobSendToQueue statement and can be also obtained using QueueJobList statement. An error occurs if the specified job_runtime_id cannot be found

 **Important note:**

- This statement is not supported in remote jobs executed on 24x7 Remote Agents. An error “Job not found” will occur if you use this statement.
- Do not confuse JobKill, JobDelete and JobDisable statements. See descriptions of these statements for more details.
- This statement is not supported in standalone JAL scripts executed from command line.

**See also:**

QueueJobList
JobSendToQueue
JobDelete
JobHold
JobRun

# JobRun

**Description:** Runs the specified job.

**Syntax:** JobRun job

| Argument | Description |
|---|---|
| Job | A string whose value is ether job ID or job name |

**Return value:** None

**Usage:** Use JobRun in your scripts to programmatically start other jobs. This allows creation of simple and complex batch jobs.
JobRun can start jobs from the active job pool only. If the specified job is not found, an error occurs.

**Important note:**

- The JobRun statement is not supported in remote jobs executed on 24x7 Remote Agents. An error “Job not found” will occur if you use this statement. However, it can be used if the remote job is executed on the 24x7 Master Scheduler.

**See also:**

JobEnable
JobGetStatus
JobRemoteRun

# JobRemoteRun

**Description:** Runs the specified job using the specified remote agent.

**Syntax:** JobRemoteRun job, agent

| Argument | Description |
|---|---|
| Job | A string whose value is ether job ID or job name |
| Agent | A string whose value is the name of 24x7 Remote Agent or 24x7 Master Scheduler as it is specified in the Remote Agent profile. |

**Return value:** None

**Usage:**
Use JobRemoteRun in your scripts to programmatically start other jobs. This allows creation of simple and complex batch jobs.

JobRemoteRun allows dynamic selection for the Remote Agent that will execute the specified job. JobRemoteRun ignores name of the default Remote Agent that may be assigned to the job in the job properties, where JobRun statement instead uses the default Remote Agent if there is any.

JobRemoteRun can start jobs from the active job pool only. If the specified job is not found, an error occurs.

**Important note:**

- The JobRemoteRun statement is not supported in remote jobs executed on 24x7 Remote Agents. An error “Job not found” will occur if you use this statement. However, it can be used if the remote job is executed on the 24x7 Master Scheduler.

**See also:**

JobEnable
JobGetStatus
JobRun
GetRemoteVariable
SetRemoteVariable

# JobSendToQueue

**Description:** Queues the specified job for later execution. The job goes to the queue specified in job properties.

**Syntax:** JobSendToQueue job, return

| Argument | Description |
|---|---|
| Job | A string whose value is ether job ID or job name |
| return | A numeric variable receiving the value of the job run-time ID |

**Return value:** None

**Usage:** Use JobSendToQueue in your scripts to programmatically start other jobs. This allows creation of simple and complex batch jobs. JobSendToQueues can start jobs from the active job pool only. If the specified job is not found, an error occurs.

Use JobSendToQueue statement to add jobs to queues and allow calling jobs to continue running not waiting for their completion.

**Important note:**

- This statement is not supported in remote jobs executed on 24x7 Remote Agents. An error “Job not found” will occur if you use this statement. However, it can be used if a remote job is executed within 24x7 Master Scheduler.
- This statement is not supported in standalone JAL scripts executed from command line.

**See also:**

QueueJobList
JobRun
JobRemoteRun

# QueueJobList

**Description:** Reports jobs being executed and waiting in job queues.

**Syntax:** QueueJobList queue_name, return

| Argument | Description |
|---|---|
| queue_name | A string whose value job queue name |
| return | A string variable receiving the output list |

**Return value:** None

**Usage:** QueueJobList returns tab-separated multi-line value describing jobs waiting and running in the specified queue. The returned value is reported in the following format

| | | | |
|---|---|---|---|
| job 1 run time ID | job 1 status | job 1 ID | job 1 process identifier |
| job 2 run time ID | job 2 status | job 2 ID | job 2 process identifier |
| job 3 run time ID | job 3 status | job 3 ID | job 3 process identifier |

…
job N run time ID  job N status  job N ID  job N process identifier

The number of returned lines matches the number of jobs currently running and waiting in the specified queue. Individual values within each line are tab separated. The returned value can be parsed and split into individual elements using GetToken statement.

**Important note:**
- This statement is not supported in remote jobs executed on 24x7 Remote Agents. However, it can be used if a remote job is executed within 24x7 Master Scheduler.
- This statement is not supported in standalone JAL scripts executed from command line.

**See also:**
JobList
JobRun
JobSendToQueue

# GetRemoteVariable

**Description:** Obtains value of the specified global variable on the specified 24x7 Remote Agent or 24x7 Master Scheduler.

**Syntax:** GetRemoteVariable agent, variable, return

| Argument | Description |
|---|---|
| Agent | A string whose value is the name of 24x7 Remote Agent or 24x7 Master Scheduler as it is specified in the Remote Agent profile. |
| Variable | A string whose value is the name of the global variable that must exists in the run-time environment of the Agent. |
| Return | A string variable that receives the returned value. |

**Return value:** Returns value of a global variable stored in the run-time environment of the Agent. The value is returned in the string format regardless of the data type of the remote variable.

**Usage:**
Use GetRemoteVariable and SetRemoteVariable statements to pass data between networked 24x7 Scheduler components. This can be used to build complex distributed jobs. This can be also used for synchronization of jobs running simultaneously on different computers.

**See also:**
Dim
Set
SetRemoteVariable
JobGetStatus
JobRemoteRun

# SetRemoteVariable

**Description:** Sets value of the specified global variable on the specified 24x7 Remote Agent or 24x7 Master Scheduler.

**Syntax:** SetRemoteVariable agent, variable, new_value

| Argument | Description |
|---|---|
| Agent | A string whose value is the name of 24x7 Remote Agent or 24x7 Master Scheduler as it is specified in the Remote Agent profile. |
| Variable | A string whose value is the name of the global variable that must exists in the run-time environment of the Agent. |
| New_Value | A constant or a variable whose value you want to store in the remote variable. The data type of the new_value must match the data type of the remote global variable. |

**Return value:** None.

**Usage:**
Use GetRemoteVariable and SetRemoteVariable statements to pass data between networked 24x7 Scheduler components. This can be used to build complex distributed jobs. This can be also used for synchronization of jobs running simultaneously on different computers.

**See also:**

Dim
Set
JobRemoteRun
GetRemoteVariable

# RemoteCopyJob

**Description:** Copies an existing job to the specified remote 24x7 Scheduler.

**Syntax:** RemoteCopyJob host, job, append

| Argument | Description |
|---|---|
| host | A string whose value is the name 24x7 Master Scheduler as it is specified in the Remote Agent profile. |
| job | A string whose value is ether source job ID or job name. |
| append | A boolean whose value indicates how you want to copy the job. Use TRUE value for "always **append**" mode and FALSE for "**replace** if exists, otherwise **append**" mode. |

| | |
|---|---|
| | If a TRUE is specified, a new job is created and appended to the remote job database. The properties of the newly created job are exactly the same as properties of the source job. The only exception is the new job ID, which must be unique and because of that it can differ from ID of the source job. If the remote database does not have a folder whose name matches the name of the folder containing the source job, then RemoteCopyJob creates a new folder in the target job database.<br><br>If a FALSE is specified, a new job is created only if there is no job with the same name and folder exists in the target job database. If either job or folder do not exist in the target job database, then the RemoteCopyJob operation proceeds just as if a TRUE value was specified. If the same job is found in the target job database, then its definition is updated to match the definition of the source job. |

**Return value:** None

**Usage:** Use this statement to programmatically replicate jobs from the 24x7 Scheduler executing the RemoteCopyJob statement to the remote 24x7 Master Scheduler.

**Important note:**

- The RemoteCopyJob statement cannot be used for copying jobs from/to 24x7 Remote Agents. It can be only used to append or replace jobs on remote 24x7 Master Schedulers.

**See also:**

RemoteJobCreate
RemoteJobModify
RemoteCopyJobFolder
RemoteCopyJobDatabase

# RemoteCopyJobFolder

**Description:** Copies job folder and all jobs contained in that folder to the specified remote 24x7 Scheduler.

**Syntax:** RemoteCopyJobFolder host, folder, append

| Argument | Description |
|---|---|
| host | A string whose value is the name 24x7 Master Scheduler as it is specified in the Remote Agent profile. |
| folder | A string whose value is source folder name. |
| append | A boolean whose value indicates how you want to copy jobs from the source folder. Use TRUE value for "always **append**" mode and FALSE for "**replace** if exists, otherwise **append**" mode.<br><br>24x7 Scheduler recursively copies jobs from the source folder as if the RemoteCopyJob statement was executed for each of them. |

| | |
|---|---|
| | If the target folder does not exists, RemoteCopyJobFolder creates it.<br><br>If the target folder exists and a FALSE is specified, the target folder is cleaned before the job copy operation begins. |

**Return value:** None

**Usage:** Use this statement to programmatically replicate folders and jobs from the 24x7 Scheduler executing the RemoteCopyJobFolder statement to the remote 24x7 Master Scheduler.

**Important note:**

- The RemoteCopyJobFolder statement cannot be used for copying jobs from/to 24x7 Remote Agents. It can be only used to append or replace jobs on remote 24x7 Master Schedulers.

**See also:**

RemoteJobCreate
RemoteJobModify
RemoteCopyJob
RemoteCopyJobDatabase

# RemoteCopyJobDatabase

**Description:** Copies alls job folders and jobs to the specified remote 24x7 Scheduler.

**Syntax:** RemoteCopyJobDatabase host, append

| Argument | Description |
|---|---|
| host | A string whose value is the name 24x7 Master Scheduler as it is specified in the Remote Agent profile. |
| append | A boolean whose value indicates how you want to copy folders and jobs from the source job database. Use TRUE value for "always **append**" mode and FALSE for "**replace** if exists, otherwise **append**" mode.<br><br>24x7 Scheduler recursively copies folders and jobs from the source database as if the RemoteCopyJobFolder statement was executed for each folder.<br><br>RemoteCopyJobDatabase creates folders that do not exist in the target database.<br><br>If a FALSE value is specified, RemoteCopyJobDatabase cleans target folders that already exist; it also deletes folders that exist in the target job database but not in the source job database. |

**Return value:** None

**Usage:** Use this statement to programmatically replicate job database from the 24x7 Scheduler executing the RemoteCopyJobDatabase statement to the remote 24x7 Master Scheduler.

**Important note:**

- The RemoteCopyJobDatabase statement cannot be used for copying jobs from/to 24x7 Remote Agents. It can be only used to append or replace jobs and folders on remote 24x7 Master Schedulers.

**See also:**

RemoteJobCreate
RemoteJobDelete
RemoteCopyJob
RemoteCopyJobFolder
RemoteCopySettings

# RemoteCopySettings

**Description:** Copies 24x7 system settings to the specified remote 24x7 Scheduler or 24x7 Remote Agent.

**Syntax:** RemoteCopySettings host, settings_group

| Argument | Description |
|---|---|
| host | A string whose value is the name 24x7 Master Scheduler as it is specified in the Remote Agent profile. |
| settings_group | A string whose value indicates which settings you want to copy. The following values are supported:<br>• "AGENT PROFILES" – use this value to copy Remote Agent profiles<br>• "DB PROFILES" – use this value to copy Database Profiles<br>• "QUEUES" – use this value to copy Queues settings<br>• "HOLIDAYS" – use this value to copy holidays table<br>• "SCRIPT LIBRARY" – use this value to copy the Script Library<br>• "SECURITY" – use this value to copy Security settings, including users and permissions<br>• "" – Use an empty string to copy all of the above |

**Return value:** None

**Usage:** Use this statement to programmatically replicate 24x7 configuration profiles and data from the 24x7 Scheduler (or 24x7 Remote Agent) executing the RemoteCopyJobSettings statement to the remote 24x7 Master Scheduler (or 24x7 Remote Agent).

**See also:**

Script Library
Job Queues
Remote agent Profiles
Database Profiles
Security
Holiday List

RemoteCopyJobDatabase

# RemoteJobCreate

**Description:** Creates a new job on the specified remote 24x7 Scheduler

**Syntax:** RemoteJobCreate host, job_data, return

| Argument | Description |
|---|---|
| host | A string whose value is the name 24x7 Master Scheduler as it is specified in the Remote Agent profile. |
| job_data | A string whose value is the job definition in **JDL format**. For examples on how to specify job definition please see available Job Templates that can be found in the Template directory. |
| return | A numeric variable that receives the returned value. |

**Return value:** Number. Returns job ID for the new job.

**Usage:** Use RemoteJobCreate in your scripts to programmatically create new remote jobs. Use RemoteJobModify statement to modify properties of the created job.

 **Important note:**

- The RemoteJobCreate statement cannot be used to create jobs on 24x7 Remote Agents. It can be only used to create jobs on remote 24x7 Master Schedulers.

**See also:**

RemoteJobDelete
RemoteJobEnable
RemoteJobModify
JobCreate

# RemoteJobDelete

**Description:** Delete an existing job on the specified remote 24x7 Scheduler

**Syntax:** RemoteJobDelete host, job

| Argument | Description |
|---|---|
| host | A string whose value is the name 24x7 Master Scheduler as it is specified in the Remote Agent profile. |

| | |
|---|---|
| job | A string whose value is ether job ID or job name. |

**Return value:** None.

**Usage:** Use RemoteJobDelete in your scripts to programmatically delete remote jobs. Use RemoteJobEnable statement to programmatically enable or disable remote jobs.

**Important note:**

- The RemoteJobDelete statement cannot be used to delete jobs on 24x7 Remote Agents. It can be only used to delete jobs on remote 24x7 Master Schedulers.

**See also:**

RemoteJobCreate
RemoteJobEnable
RemoteJobModify
JobDelete

# RemoteJobDescribe

**Description:** Describes properties of an existing job on the specified remote 24x7 Scheduler.

**Syntax:** RemoteJobDescribe host, job, property, value

| Argument | Description |
|---|---|
| host | A string whose value is the name 24x7 Master Scheduler as it is specified in the Remote Agent profile. |
| job | A string whose value is ether job ID or job name. |
| property | A string whose value is the property that you want to describe. See Job Properties topic for list and description of available properties |
| value | A string variable that receives the returned value. |

**Return value:** A string whose value matches the value for the property. The returned value is always converted to a string.

**Usage:** Use this statement to retrieve job definitions, job schedules and triggers programmatically. For example, you can use RemoteJobDescribe to create customizable job monitors.

**Note:**

- The RemoteJobDescribe statement cannot be used with 24x7 Remote Agents. It can be only used to obtain properties of jobs setup on remote 24x7 Master Schedulers.

**See also:**

RemoteJobCreate
RemoteJobDelete
RemoteJobModify

JobDescribe

# RemoteJobEnable

**Description:** Enables or disables an existing job on the specified remote 24x7 Scheduler

**Syntax:** RemoteJobEnable host, job, state

| Argument | Description |
|---|---|
| host | A string whose value is the name 24x7 Master Scheduler as it is specified in the Remote Agent profile. |
| job | A string whose value is ether job ID or job name. |
| state | A boolean whose value is the new job state. Specify TRUE to enable the job or FALSE to disable it. |

**Return value:** None.

**Usage:** Use RemoteJobEnable in your scripts to programmatically enable or disable remote jobs. If the specified job is not found, an error occurs.

**Important Notes:**

- The RemoteJobEnable statement cannot be used to enable/disable jobs on 24x7 Remote Agents. It can be only used to change state of jobs on remote 24x7 Master Schedulers.
- To physically remove a job from the job database use RemoteJobDelete, JobDelete, 24x7 GUI or one of the supported external interfaces.

**See also:**

RemoteJobCreate
RemoteJobDelete
RemoteJobModify
JobEnable

# RemoteJobList

**Description:** Returns comma-separated list of job IDs in the active job database on the specified remote host.

**Syntax:** RemoteJobList host, folder, return

| Argument | Description |
|---|---|
| host | A string whose value is the name 24x7 Master Scheduler as it is specified in the Remote Agent profile. |

| | |
|---|---|
| folder | A string whose value is the name of the folder in which you want to list jobs. Specify an empty string ("") for the folder to obtain a list of all job IDs in the job database. |
| return | A string variable that receives the returned value. |

**Return value:** A string containing comma-separated list of job IDs.

**Usage:** The RemoteJobList statement can be used to obtain a list of jobs setup on a remote 24x7 Master Scheduler.

**Tip:**
- You can use GetToken and LoopWhile statements to parse the returned job list.

**See also:**

JobRemoteRun
RemoteCopyJob
RemoteJobDescribe
JobList
Job Properties

# RemoteJobModify

**Description:** Modifies properties of an existing job on the specified remote 24x7 Scheduler.

**Syntax:** RemoteJobModify host, job, property, new_value

| Argument | Description |
|---|---|
| host | A string whose value is the name 24x7 Master Scheduler as it is specified in the Remote Agent profile. |
| job | A string whose value is ether job ID or job name. |
| property | A string whose value is the property that you want to modify. See **Job Properties** topic for list and description of available properties |
| new_value | A string whose value is the new value for the property |

**Return value:** None

**Usage:** Use this statement to modify job definitions, job schedules and triggers programmatically. For example, you can use RemoteJobModify to reschedule jobs for a later run when the exact start time is unknown in design-time.

**Important Notes:**
- The RemoteJobModify statement cannot be used with 24x7 Remote Agents. It can be only used to change properties of jobs setup on remote 24x7 Master Schedulers.
- Modified jobs are updated immediately in both places: the active job pool and the job database. 24x7 Scheduler internally triggers "save" command after every change.
- Side effect: On "save," all disabled jobs are purged from the active job pool. However, disabled jobs are not removed from the job database. To physically remove a job from the job database use RemoteJobDelete statement or JobDelete statement or 24x7 GUI or one of supported external interfaces.

**See also:**

RemoteJobCreate
RemoteJobDelete
RemoteJobDescribe
JobModify

# Web statements

## Ping

**Description:** Checks whether the specified remote computer is online. It simulates PING utility that used to test and debug a network by sending out a data packet and waiting for a response.

**Syntax:** Ping computer, return

| Argument | Description |
|---|---|
| Computer | A string whose value is either name of the remote computer as specified in your DNS Server (for example, www.softtreetech.com, ftp.softtreetech.com) or the IP number of the remote computer in ASCII dotted-decimal format (for example, 11.0.1.45) |
| Return | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE if the remote computer can be found (alive), and FALSE otherwise.

**Usage:** Use Ping to test TCP/IP connectivity and check for availability of a remote computer before making connection to that computer. You can also use Ping in various monitoring jobs of "Server Alive" kind.

**Important Notes:**

- Ping statement is available only if you have Winsock 2 installed on your system. To verify that check if you have WS2_32.DLL in your WINDONS\SYSTEM of WINNT/SYSTEM32 directory. Most Windows 98/Me/NT/2000/XP/2003/VISTA/2008/7 systems should have Winsock 2 files installed by default. If you are running Windows 95 and don't have Winsock 2 installed on your system you can freely obtain it at the following location http://www.microsoft.com/windows/downloads/bin/W95ws2setup.exe.
- Ping statement is available only if the TCP/IP protocol has been installed.

**Tips:**

- Turn of **Tracing** feature to see additional status messages returned by the Ping statement.
- If you see "Request timed out" message in the trace, verify that the host IP address is correct, that the host is operational, and that all the gateways (routers) between this computer and the host are operational.
- To test host name resolution by using the Ping statement, Ping the desired host using its host name. If the Ping fails with an "Unknown host" message in the trace, verify that the host name is correct and that the host name can be resolved by your DNS server.

**See also:**

PingPort

FTP Statements
Web statements

# PingPort

**Description:** Checks whether the specified port (e.g. network service) is responsive on the specified remote computer.

**Syntax:** PingPort computer, port, return

| Argument | Description |
|---|---|
| Computer | A string whose value is either name of the remote computer as specified in your DNS Server (for example, www.softtreetech.com, ftp.softtreetech.com) or the IP number of the remote computer in ASCII dotted-decimal format (for example, 11.0.1.45) |
| Port | A number whose value is the port number that you want to check. |
| Return | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE if the Port on remote computer is responding (alive), and FALSE otherwise.

**Usage:** Use PingPort to test various network services (for example, FTP) before using them. You can also use PingPort in various monitoring jobs of "Service Alive" kind.

The following table lists default port numbers for most commonly used network services.

| Service Name | Port |
|---|---|
| daytime | 13 |
| netstat | 15 |
| FTP | 21 |
| telnet | 23 |
| SMTP | 25 |
| DNS | 53 |
| finger | 79 |
| HTTP | 80 |
| rlogin | 513 |
| rsh | 514 |
| UUCP | 540 |
| klogin | 543 |
| krcmd, kshell | 544 |

**Important Notes:**

- PingPort statement is available only if you have Winsock 2 installed on your system. To verify that check if you have WS2_32.DLL in your WINDONS\SYSTEM of WINNT/SYSTEM32 directory. Most Windows 98/Me/NT/2000/XP/2003/VISTA/2008/7 systems should have Winsock 2 files installed by default. If you are running Windows 95 and don't have Winsock 2 installed on your system you can freely obtain it at the following location http://www.microsoft.com/windows/downloads/bin/W95ws2setup.exe.
- PingPort statement is available only if the TCP/IP protocol has been installed.

**Tips:**

- Turn of **Tracing** feature to see additional status messages returned by the Ping statement.
- If you see "Request timed out" message in the trace, verify that the host IP address is correct, that the host is operational, and that all the gateways (routers) between this computer and the host are operational.
- To test host name resolution by using the Ping statement, Ping the desired host using its host name. If the Ping fails with an "Unknown host" message in the trace, verify that the host name is correct and that the host name can be resolved by your DNS server.

**See also:**

Ping

FTP Statements

Web statements

# WebConfig

**Description:** Set various parameters for the subsequent Internet operations.

**Syntax:** WebConfig property, new_value

| Argument | Description |
|---|---|
| Property | A string whose value is the name of the property for the Internet session that you want to change. The following properties are supported:<br>• "PROXY" |
| New_value | • A string whose value is the new value for the property that you want to change. |

**Return value:** None.

**Usage:** Use WebConfig statement to configure your Internet connection before accessing file on the Internet using WebGetFile and WebPostData statements. Both parameter name and new value are case insensitive.

"PROXY" setting consists of 4 parts separated by commas: proxy server name (or IP address), proxy server port, your user ID and password required for proxy authentication. If your proxy server does not require authentication, do not specify user ID and password. If you use different proxies for different protocols such as HTTP and FTP, call WebConfig with different parameters before using WebGetFile, WebPostData and other Web statements for a different protocol.

**Example:**
*WebConfig "PROXY", "127.01.01.3,8080,myname,mypassword"*

**See also:**

WebPostData
WebPostDataWithLogin
WebGetFile
WebGetDatawithLogin

# WebGetFile

**Description:** Downloads data from the specified URL and saves it to the specified file. The downloaded file can be of any type, including but not limited to static HTML pages, dynamic HTML pages generated by web server scripts, image files, PDF files, files of any other type sent by the web server.

**Syntax:** WebGetFile url, file

| Argument | Description |
|---|---|
| URL | A string whose value is the URL ( Internet standard Uniform Resource Locator, e.g. Web address ) that points to the file that you want to download |
| File | A string whose value is the name of the local file in which you want to save returned data |

**Return value:** None.

**Usage:** The specified URL can point to the file of any downloadable type, including HTML files, ASCII files, image files, and other binary files. If the specified URL points to a Internet server executable file such as CGI, ASP, DLL, and other, WebGetFile returns the data source that is created as a result of processing on the Internet server. You can use WebGetFile statement for downloading various Internet files.

If the web site specified by URL requires user authentication, use the following URL syntax:
<username>:<password>@<protocol>://<server name>/<...path and file name>
Example: *myname:mypassword@http://www.mycompany.com/webstats/access.log*

If you connect to the Internet via proxy server, use WebConfig statement to specify connection parameters required by your proxy server.

**Note:**

- WebGetFile statement can use HTTP and FTP protocols when accessing files on the Web. If you omit protocol name in the URL specification then HTTP protocol is used by default.

**See also:**

WebConfig
WebGetDataWithLogin
WebPostData
WebPostDataWithLogin
WebStripHTMLTags

FTPGetFile
WebOpenPage

# WebGetPageHTML

**Description:** Retrieves page source for the specified URL. Page source can be HTML file or a file of other type. This statement is provided for compatibility with previous versions of JAL script engine. New JAL script engine jobs should use the more powerful **WebGetFile** statement and **WebGetDataWithLogin.**

**Syntax:** WebGetPageHTML url, file

| Argument | Description |
|---|---|
| url | A string whose value is the URL whose source data you want to retrieve |
| file | A string whose value is the name of the local file in which you want to save returned data |

**Return value:** None.

**Usage:** The specified URL can point to a file of any downloadable type, including HTML files, ASCII files, image files, and other binary files. If the specified URL points to a Internet server executable file such as CGI, ASP, DLL, and other, WebGetPageHTML returns the data source that is created as a result of processing on the Internet server. You can use WebGetPageHTML statement for downloading various Internet files.

 **Note:**

- WebGetPageHTML statement uses HTTP protocol for any type of file transfer.
- WebGetPageHTML statement does not support proxy connections. It is provided only for compatibility with previous versions of 24x7 Scheduler. New 24x7 Scheduler jobs should use the more powerful **WebGetFile** statement.

**See also:**

FTPGetFile
WebOpenPage

# WebGetDataWithLogin

**Description:** Performs HTTP POST and then immediately HTTP GET, allowing a job to submit a request through CGI, NSAPI, or ISAPI call to a login form on a remote web site, and then, in the same authenticated user session download data from the same or a different location through second CGI, NSAPI, or ISAPI call.

**Syntax:** WebGetDataWithLogin login_url, login_data, error_tokens, data_url, output_format, file, logout_url

| Argument | Description |
|---|---|
| Login_URL | A string whose value is the URL ( Internet standard Uniform Resource Locator, e.g. Web address ) of the login form which can be used to login to the web site. |
| Login_Data | A string whose value is the data to be posted to the specified Login_URL |
| Error_Tokens | A string containing comma-separated substrings that you want to check in the results returned by the login form in to verify whether the login succeeded or not. If you do not want to check the login form output, specify an empty string for this argument. Note that the search for substrings is case-insensitive. |
| Data_URL | A string whose value is the URL ( Internet standard Uniform Resource Locator, e.g. Web address ) to get data from |
| Output_format | A string whose value describes the expected data format for the data returned by your web server after successful GET request. The value is optional. For most results, you can specify an empty string for Output_format argument and let the script engine to handle data using default rules. If you need to specify a value for the output format, use MIME content-type names described in RFC 2387 standard. See common content-type names in the following Usage section. |
| File | A string whose value is the name of the local file in which you want to save the returned web server response and data |
| Logout_URL | A string whose value is the URL ( Internet standard Uniform Resource Locator, e.g. Web address ) of the logout page which can be used to logout from the web site. The Logout_URL argument value is optional. In case logout is not required, specify an empty string as value for this argument. |

**Return value:** None.

**Usage:** Use this statement to download data from a remote web site that requires prior from-based user authentication. You can also use the **Web Automation Wizard** to quickly build HTTP GET automation scripts

A typical web login form contains several fields such as user name and password, and in case of an invalid login, it outputs some errors. The Error_Tokens argument can be used to automate error checking for the login. Basically, after posting login data to the login form, WebPostDataWithLogin searches text of the web server response for substrings specified in the Error_Tokens argument. If any specified substring is found, the statement raises run-time error, and the GET operation aborts. The search for substrings is not case sensitive.

The results returned after the GET operation can be of any downloadable type, including HTML files, ASCII files, image files, and other types of text and binary files. The returned data is saved in the specified target File. The target file is overwritten on every run.

Here is a list of popular content-type values that can be specified for the Output_format argument:
An empty string – this allows the script engine to recognize and automatically handle the output format.
text/plain – any plain text result. Typically, this format should be handled automatically.
text/html – HTML formatted data. Typically, this format should be handled automatically.
text/xml – XML formatted data. Typically, this format should be handled automatically.
image/jpeg- JPG image files.
image/png – PNG image files.
image/bmp – BMP image files.
image/gif – GIF image files.
audio/mpeg – MPEG audio files.
audio/mpeg3 – MPEG 3 audio files.

video/avi – AVI video files.
video/mpeg – MPEG video files.
application/zip – Zip compressed files.
application/x-zip – Zip compressed files.
application/x-gzip - Gzip compressed files.
application/pdf – Adobe Acrobat files.
application/excel – Microsoft Excel files.
application/msword – Microsoft Word files.
… many other – refer to your web application documentation for details on the supported data formats.

If the target web site does not require form-based user authentication, use WebGetFile statement.

If you connect to the Internet via proxy server, use WebConfig statement to specify connection parameters required by your proxy server.

**See also:**

WebConfig
WebGetFile
WebGetDataWithLogin
WebPostData
WebHTMLEncode
WebURLEncode
FTPPutFile
WebOpenPage

# WebOpenPage

**Description:** Opens default Web browser, displaying the specified URL.

**Syntax:** WebOpenPage url

| Argument | Description |
|---|---|
| url | A string whose value is the URL you want to open in the default browser |

**Return value:** None.

**See also:**

WebGetPageHTML

# WebPostData

**Description:** Performs an HTTP POST, allowing a job to upload data to a remote web site through CGI, NSAPI, or ISAPI call.

**Syntax:** WebPostData url, data, file

| Argument | Description |
|---|---|
| URL | A string whose value is the URL ( Internet standard Uniform Resource Locator, e.g. Web address ) to post data to |
| Data | A string whose value is the data to be posted to the specified URL |
| File | A string whose value is the name of the local file in which you want to save the returned web server response |

**Return value:** None.

**Usage:** Use this statement to invoke a CGI, NSAPI, or ISAPI function on a web server, in other words, to post data to a remote web site. The data should be sent in URL-encoded format as specified in RFC1738 standard http://www.ietf.org/rfc/rfc1738.txt. Use WebURLEncode statement to encode the data before uploading it to the target site using WebPostData or WebPostDataWithLogin statements. You can also use the **Web Automation Wizard** to quickly build HTTP POST automation scripts.

The results returned after the POST operation can be of any downloadable type, including HTML files, ASCII files, image files, and other types of text and binary files. This data is saved in the specified target File. The target file is overwritten on every run.

If the target web site specified by URL requires basic type of user authentication, use the following URL syntax:
<username>:<password>@<protocol>://<server name>/<...path and file name>
Example: *myname:mypassword@http://www.mycompany.com/folder/input-form.php*

Use WebPostDataWithLogin statement in case the target site requires form-based authentication before any data can be posted to the site.

If you connect to the Internet via proxy server, use WebConfig statement to specify connection parameters required by your proxy server.

**See also:**

- WebConfig
- WebGetDataWithLogin
- WebGetFile
- WebPostDataWithLogin
- WebHTMLEncode
- WebURLEncode
- FTPPutFile
- WebOpenPage

# WebPostDataWithLogin

**Description:** Performs double HTTP POST, allowing a job to submit a request through CGI, NSAPI, or ISAPI call to a login form on a remote web site, and then, in the same authenticated user session upload data to the same or a different location through second CGI, NSAPI, or ISAPI call.

**Syntax:** WebPostDataWithLogin login_url, login_data, error_tokens, data_url, data, output_format, file, logout_url

| Argument | Description |
|---|---|
| Login_URL | A string whose value is the URL (Internet standard Uniform Resource Locator, e.g. Web address) of the login form which can be used to login to the web site. |
| Login_Data | A string whose value is the data to be posted to the specified Login_URL |
| Error_Tokens | A string containing comma-separated substrings that you want to check in the results returned by the login form in to verify whether the login succeeded or not. If you do not want to check the login form output, specify an empty string for this argument. Note that the search for substrings is case-insensitive. |
| Data_URL | A string whose value is the URL (Internet standard Uniform Resource Locator, e.g. Web address) to post data to |
| Data | A string whose value is the data to be posted to the specified Data_URL |
| Output_format | A string whose value describes the expected data format for the data returned by your web server after successful POST request. The value is optional. For most results, you can specify an empty string for Output_format argument and let the script engine to handle data using default rules. If you need to specify a value for the output format, use MIME content-type names described in RFC 2387 standard. See common content-type names in the following Usage section. |
| File | A string whose value is the name of the local file in which you want to save the returned web server response and data |
| Logout_URL | A string whose value is the URL (Internet standard Uniform Resource Locator, e.g. Web address) of the logout page, which can be used to logout from the web site. The Logout_URL argument value is optional. In case logout is not required, specify an empty string as value for this argument. |

**Return value:** None.

**Usage:** Use this statement to invoke a CGI, NSAPI, or ISAPI function on a web server, in other words, to post data to a remote web site that requires prior from-based user authentication. The data should be sent in URL-encoded format as specified in RFC1738 standard http://www.ietf.org/rfc/rfc1738.txt. Use WebURLEncode statement to encode the data before uploading it to the target site using WebPostData or WebPostDataWithLogin statements. You can also use the **Web Automation Wizard** to quickly build HTTP POST automation scripts

A typical web login form contains several fields such as user name and password, and in case of an invalid login, it outputs some errors. The Error_Tokens argument can be used to automate error checking for the login. Basically, after posting login data to the login form, WebPostDataWithLogin searches text of the web server response for

substrings specified in the Error_Tokens argument. If any specified substring is found, the statement raises run-time error, and the GET operation aborts. The search for substrings is not case sensitive.

The results returned after the POST operation can be of any downloadable type, including HTML files, ASCII files, image files, and other types of text and binary files. The returned data is saved in the specified target File. The target file is overwritten on every run.

Here is a list of popular content-type values that can be specified for the Output_format argument:
An empty string – this allows script engine to recognize and automatically handle the output format.
text/plain – any plain text result. Typically, this format should be handled automatically.
text/html – HTML formatted data. Typically, this format should be handled automatically.
text/xml – XML formatted data. Typically, this format should be handled automatically.
image/jpeg- JPG image files.
image/png – PNG image files.
image/bmp – BMP image files.
image/gif – GIF image files.
audio/mpeg – MPEG audio files.
audio/mpeg3 – MPEG 3 audio files.
video/avi – AVI video files.
video/mpeg – MPEG video files.
application/zip – Zip compressed files.
application/x-zip – Zip compressed files.
application/x-gzip - Gzip compressed files.
application/pdf – Adobe Acrobat files.
application/excel – Microsoft Excel files.
application/msword – Microsoft Word files.
… many other – refer to your web application documentation for details on the supported data formats.

If the target web site does not require form-based user authentication, use WebPostData statement.

If you connect to the Internet via proxy server, use WebConfigstatement to specify connection parameters required by your proxy server.

**See also:**

- WebConfig
- WebGetFile
- WebGetDataWithLogin
- WebPostData
- WebHTMLEncode
- WebURLEncode
- FTPPutFile
- WebOpenPage

# WebHTMLEncode

**Description:** Applies HTML-encoding to the specified text string. This statement provides convenient means for encoding data to be used in HTML and XML files

**Syntax:** WebHTMLEncode data, return

| Argument | Description |
|---|---|
| Data | A string value that you want to encode. |

| | |
|---|---|
| Return | A string variable that receives the returned value |

**Return value:** HTML encoded string.

**Usage:** The encoding replaces characters that cannot be used in HTML/XML data with their encoded values. For more details see "Hypertext Markup Language - 2.0" specification described in RFC 1866 standard.

Here is a partial list of characters that are encoded by WebHTMLEncode statement.

| CHARACTER | ENCODED | REF | CHARACTER DESCRIPTION |
|---|---|---|---|
| & | &amp; | &#38; | Ampersand |
| < | &lt; | &#60; | Less than |
| > | &gt; | &#62; | Greater than |
| Other special characters… | | | |

**See also:**
WebURLEncode
WebGetDataWithLogin
WebPostData
WebPostDataWithLogin

# WebURLEncode

**Description:** Applies URL-encoding to the specified text string. This statement provides convenient means for encoding strings to be used in a data for posting to web form.

**Syntax:** WebURLEncode data, return

| Argument | Description |
|---|---|
| Data | A string value that you want to encode. |
| Return | A string variable that receives the returned value |

**Return value:** URL encoded string.

**Usage:** The encoding is performed according to RFC 1738 standard.

**See also:**
WebHTMLEncode
WebGetDataWithLogin
WebPostData
WebPostDataWithLogin

## WebStripHTMLTags

**Description:** Removes HTML and XML tags from the specified string. This statement provides convenient means for parsing HTML and XML data.

**Syntax:** WebStripHTMLTags data, return

| Argument | Description |
|---|---|
| Data | An HTML or XML string value that you want to parse. |
| Return | A string variable that receives the returned value |

**Return value:** [lain text string.

**Usage:** This statement removes matching pairs of HTML and XML tags, for example, <td> </td>, as well as singular HTML tags such as <hr>, </br>, <br> and other. Specified data must be a well formed HTML or XML string. Malformed data may lead to incorrect results.

 **Note:**

- HTML comments in <!-- .. --> format are removed along with HTML tags, including all text between start and end comment tags.

**See also:**

WebURLEncode
WebHTTPEncode
WebGetDataWithLogin

# Logical statements

## And

**Description:** Obtains the result of logical operation AND.

**Syntax:** And boolean1, boolean2, return

| Argument | Description |
|---|---|
| boolean1 | A boolean value you want to compare with value in boolean2 |
| boolean2 | A boolean value you want to compare with value in boolean1 |
| return | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns result of logical AND operation. boolean1 AND boolean2 is true if both are true. boolean1 AND boolean2 is false if either is false. See the table below for returned values.

| Boolean1 | Boolean2 | Result |
|---|---|---|
| TRUE | TRUE | TRUE |
| FALSE | FALSE | FALSE |
| TRUE | FALSE | FALSE |
| FALSE | TRUE | FALSE |

## IsDate

**Description:** Tests whether a string value is a valid date.

**Syntax:** IsDate ( string, result )

| Argument | Description |
|---|---|
| String | A string whose value you want to test to determine whether it is a valid date |
| result | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE if string is a valid date and FALSE if it is not.

**Note:**
Valid dates in strings can include any combination of day (1 to 31), month (1 to 12 or the name or abbreviation of a month), and year (2 or 4 digits). 24x7 Scheduler assumes a 4-digit number is a year. Leading zeros are optional for month and day. The month, whether a name, an abbreviation, or a number, must be in the month location specified in the system setting for a date's format. If you do not know the system setting, use the standard data type date format yyyy-mm-dd.

## IsDateBetween

**Description:** Tests whether a date value is between specified start and end dates.

**Syntax:** IsDateBetween ( testdate, startdate, enddate, result )

| Argument | Description |
|---|---|
| testdate | A date whose value you want to test to determine whether it is between startdate and enddate |
| startdate | A date whose value is lower limit of the specified time interval |
| enddate | A date whose value is upper limit of the specified time interval |
| result | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE if testdate is equal or later than startdate and equal or erlier than enddate, and returns FALSE if it is not.

# IsEqual

**Description:** Compares two values and returns True if the first value is equal the second one.

**Syntax:** IsEqual a, b, return

| Argument | Description |
|---|---|
| a | A value that you want to compare with b |
| b | A value against which you want compare value a |
| return | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE when a = b, and FALSE otherwise

**Usage:** The data types of value a and b must match each other.

**See also:**

IsEqual
IsGreater
IsGreaterOrEqual
IsLess
IsLessOrEqual

# IsGreater

**Description:** Compares two values and returns True if the first value is greater then the second one.

**Syntax:** IsGreater a, b, return

| Argument | Description |
|---|---|
| a | A value that you want to compare with b |
| b | A value against which you want compare value a |
| return | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE when a > b, and FALSE otherwise

**Usage:** The data types of value a and b must match each other. All data types supported except boolean.

**See also:**

IsEqual
IsGreater
IsGreaterOrEqual
IsLess
IsLessOrEqual

## IsGreaterOrEqual

**Description:** Compares two values and returns True if the first values is greater or equal the second one.

**Syntax:** IsGreaterOrEqual a, b, return

| Argument | Description |
|---|---|
| a | A value that you want to compare with b |
| b | A value against which you want compare value a |
| return | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE when a >= b, and FALSE otherwise

**Usage:** The data types of value a and b must match each other. All data types supported except boolean.

**See also:**

IsEqual
IsGreater
IsGreaterOrEqual
IsLess
IsLessOrEqual

## IsHoliday

**Description:** Tests whether the specified date falls on a holiday.

**Syntax:** IsHoliday ( testdate, result )

| Argument | Description |
|---|---|
| testdate | A date whose value you want to test to determine whether it is a holiday |
| result | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE if testdate is a holiday and FALSE if it is not.

**Note:**
You use Tools/Holidays menu command to maintain list of holidays and other exception dates for the 24x7 Scheduler.

# IsLess

**Description:** Compares two values and returns True if the first value is less then the second one.

**Syntax:** IsLess a, b, return

| Argument | Description |
|---|---|
| a | A value that you want to compare with b |
| b | A value against which you want compare value a |
| return | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE when a < b, and FALSE otherwise

**Usage:** The data types of value a and b must match each other. All data types supported except boolean.

**See also:**

IsEqual
IsGreater
IsGreaterOrEqual
IsLess
IsLessOrEqual

# IsLessOrEqual

**Description:** Compares two values and returns True if the first value is less or equal the second one.

**Syntax:** IsLessOrEqual a, b, return

| Argument | Description |
|---|---|
| a | A value that you want to compare with b |
| b | A value against which you want compare value a |
| return | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE when a <= b, and FALSE otherwise

**Usage:** The data types of value a and b must match each other. All data types supported except boolean.

**See also:**

IsEqual
IsGreater
IsGreaterOrEqual
IsLess
IsLessOrEqual

## IsNumber

**Description:** Reports whether the value of a string is a number.

**Syntax:** IsNumber ( string, result )

| Argument | Description |
|---|---|
| string | A string whose value you want to test to determine whether it is a valid number |
| result | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE if string is a valid number and FALSE if it is not.

## IsTime

**Description:** Reports whether the value of a string is a valid time value.

**Syntax:** IsTime (timevalue, result )

| Argument | Description |
|---|---|
| timevalue | A string whose value you want to test to determine whether it is a valid time |
| result | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE if timevalue is a valid time and FALSE if it is not.

# IsTimeBetween

**Description:** Tests whether a time value is between specified start and end times.

**Syntax:** IsTimeBetween ( testtime, starttime, endtime, result )

| Argument | Description |
|---|---|
| testdate | A time whose value you want to test to determine whether it is between starttime and endtime |
| starttime | A time whose value is lower limit of the specified time interval |
| endtime | A time whose value is upper limit of the specified time interval |
| result | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE if testtime is equal or later than starttime and equal or earlier than endtime, and returns FALSE if it is not.

# IsWeekday

**Description:** Tests whether the specified date falls on a weekday.

**Syntax:** IsWeekday ( testdate, result )

| Argument | Description |
|---|---|
| Testdate | A date whose value you want to test to determine whether it is a weekday |
| result | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE if testdate is a weekday and FALSE if it is not.

**Note:**
Weekdays are days from Monday through Friday.

# IsWeekend

**Description:** Tests whether the specified date falls on a weekend.

**Syntax:** IsWeekend ( testdate, result )

| Argument | Description |
|---|---|
| Testdate | A date whose value you want to test to determine whether it is a weekend |
| result | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE if testdate is a weekend and FALSE if it is not.

**Note:**
Weekends include the 2 following days: Saturday and Monday.

# Not

**Description:** Obtains the result of logical operation NOT.

**Syntax:** Not boolean, return

| Argument | Description |
|---|---|
| boolean | The boolean value you want to negate |
| return | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns result of logical negation operation. See the table below for returned values.

| Boolean | Result |
|---|---|
| TRUE | FALSE |
| FALSE | TRUE |

# NotEqual

**Description:** Compares two values and returns True if the first value is not equal the second one.

**Syntax:** NotEqual a, b, return

| Argument | Description |
|---|---|
| a | A value that you want to compare with b |
| b | A value against which you want compare value a |
| return | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE when a <> b, and FALSE otherwise

**Usage:** The data types of value a and b must match each other.

**See also:**

IsEqual
IsGreater
IsGreaterOrEqual
IsLess
IsLessOrEqual

## Or

**Description:** Obtains the result of logical operation OR.

**Syntax:** Or boolean1, boolean2, return

| Argument | Description |
|---|---|
| boolean1 | A boolean value you want to compare with value in boolean2 |
| boolean2 | A boolean value you want to compare with value in boolean1 |
| return | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns result of logical OR operation. boolean1 OR boolean2 is true if either is true or both are true. boolean1 OR boolean2 is false only if both are false. See the table below for returned values.

| Boolean1 | Boolean2 | Result |
|---|---|---|
| TRUE | TRUE | TRUE |
| FALSE | FALSE | FALSE |
| TRUE | FALSE | TRUE |
| FALSE | TRUE | TRUE |

# Log statements

## LogAddMessage

**Description:** Writes an entry at the end of the Windows NT (NT/2000/XP/2003/VISTA/2008/7) application event log..

**Syntax:** LogAddMessage severity, message

| Argument | Description |
|---|---|
| severity | A string that is the severity of the message you want to add to the system application event log. It can be one of the following:<br>• “ERROR"<br>• “WARNING"<br>• “INFO" |
| message | A string that is the message you want to add to the system application event log. |

**Usage:** This statement can be used on Windows NT (NT/2000/XP/2003/VISTA/2008/7) systems only. You can use the Windows NT System Event Viewer to read event log messages.

 **Note:**
The 24x7 Scheduler maintains its own event log that is different from the system event log. LogAddMessage does not affect 24x7 Scheduler event log.

**See also:**
LogAddMessageEx
LogSearch

# LogAddMessageEx

**Description:** Adds new record to the 24x7 Scheduler event log.

**Syntax:** LogAddMessageEx severity, job_id, job_name, message

| Argument | Description |
|---|---|
| severity | A string that is the severity of the message you want to add to the 24x7 Scheduler event log. It can be one of the following:<br>• “ERROR”<br>• “WARNING”<br>• “INFO” |
| job_id | A number whose value should be ether ID of the job to which this message belongs or 0 to indicate that the message has global scope. |
| job_name | A string whose value should be ether name of the job to which this message belongs or some generic name like “24x7 Scheduler” to indicate that the message has global scope. |
| message | A string that is the message you want to add to the 24x7 Scheduler event log. |

**Usage:** LogAddMessageEx. writtes new record to the 24x7 Scheduler event log. If parallel logging to the Windows NT (NT/2000/XP/2003/VISTA/2008/7) event log enabled, LogAddMessageEx. also writes an entry at the end of the Windows NT application event log.

**Note:**
24x7 Scheduler does not verify job_id and job_name values. This is your responsibility to specify the correct values.

**See also:**
LogAddMessage
LogSearch

# LogBackup

**Description:** Saves the specified Windows NT (NT/2000/XP/2003/VISTA/2008/7) event log to a file.

**Syntax:** LogBackup log_name, file_name

| Argument | Description |
|---|---|
| log_name | A string that is the name of the Windows NT (NT/2000/XP/2003/VISTA/2008/7) event log that you want to backup. For example, it can be one of the following:<br>• “Application"<br>• "Security"<br>• "System” |
| file_name | A string that is the name of the backup file. |

**Usage:** This statement can be used on Windows NT (NT/2000/XP/2003/VISTA/2008/7) systems only. Use it to periodically backup Windows NT (NT/2000/XP/2003/VISTA/2008/7) event logs for historical auditing purposes.

**Note:**
The 24x7 Scheduler maintains its own event log that is different from the system event log. LogBackup does not affect 24x7 Scheduler event log. To backup 24x7 Scheduler's event log, use FileCopy statement to backup SCHEDULE.LOG file.

**See also:**
LogAddMessageEx
LogClear
LogSearch

# LogClear

**Description:** Clears the specified Windows NT (NT/2000/XP/2003/VISTA/2008/7) event log.

**Syntax:** LogClear log_name

| Argument | Description |
|---|---|
| log_name | A string that is the name of Windows NT (NT/2000/XP/2003/VISTA/2008/7) event log that you want to truncate. For example, it can be one of the following:<br>• “Application"<br>• "Security"<br>• "System” |

**Usage:** This statement can be used on Windows NT (NT/2000/XP/2003/VISTA/2008/7) systems only. You normally use this statement after your backup the event log using LogBackup.statement.

**Note:**
The 24x7 Scheduler maintains its own event log that is different from the system event log. LogClear does not affect 24x7 Scheduler event log. To clear 24x7 Scheduler's event log use FileDelete statement to delete SCHEDULE.LOG file.

**See also:**
LogAddMessageEx
LogBackup
LogSearch

# LogFileSize

**Description:** Retrieves current size of the specified Windows NT (NT/2000/XP/2003/VISTA/2008/7) event log and percent of free space left in that file.

**Syntax:** LogFileSize log_name, size, pct_free

| Argument | Description |
|---|---|
| log_name | A string that is the name of the Windows NT (NT/2000/XP/2003/VISTA/2008/7) event log that you want to query. For example, it can be one of the following:<br>• “Application"<br>• "Security"<br>• "System” |
| Size | A numeric variable that receives the returned value for the file size (in bytes). |
| pct_free | A numeric variable that receives the returned value for the percent of free space left in the log file (as compared to the maximum log file size) |

**Usage:** This statement can be used on Windows NT (NT/2000/XP/2003/VISTA/2008/7) systems only. Use it to monitor Windows NT (NT/2000/XP/2003/VISTA/2008/7) event logs and then use LogClear and LogBackup statements to backup and truncate these logs before they reach their maximum allowed size and start overwriting old event log data.

**Note:**
The 24x7 Scheduler maintains its own event log that is different from the system event log. To get the size of 24x7 Scheduler's event log, use FileSize statement to retrieve the size of SCHEDULE.LOG file.

**See also:**
LogAddMessageEx
LogClear
LogBackup
FileSize

# LogRecordCount

**Description:** Retrieves current number of records in the specified Windows NT (NT/2000/XP/2003/VISTA/2008/7) event log.

**Syntax:**LogRecordCount log_name, count

| Argument | Description |
|---|---|
| log_name | A string that is the name of the Windows NT (NT/2000/XP/2003/VISTA/2008/7) event log that you want to query. For example, it can be one of the following:<br>• "Application"<br>• "Security"<br>• "System" |
| count | A numeric variable that receives the returned value |

**Usage:** This statement can be used on Windows NT (NT/2000/XP/2003/VISTA/2008/7) systems only. Use it to monitor Windows NT (NT/2000/XP/2003/VISTA/2008/7) event logs and then use LogClear and LogBackup statements to backup and truncate these logs before they reach their maximum allowed size and start overwriting old event log data.

**Note:**
The 24x7 Scheduler maintains its own event log that is different from the system event log.

**See also:**
LogFileSize
LogClear
LogBackup
FileSize

# LogSearch

**Description:** Search entries in the Windows NT (NT/2000/XP/2003/VISTA/2008/7) event log.

**Syntax:** LogSearch search_system, search_warnings, start_time, return

| Argument | Description |
|---|---|
| search_system | A boolean that indicates whether you want to search the system event log or the application event log. Specify TRUE for the system log or FALSE for the application event log. |
| search_warnings | A boolean that indicates whether you want to search for messages having severity either ERROR or WARNING. Specify TRUE to search for both types or FALSE to search ERRORS only. |
| start_time | A date-time value that is the earliest time of the messages that you want to search. Use this parameter as a “filter" for the event log. 24x7 Scheduler will search only messages that were logged since the start_time. |
| Return | A string variable that receives the returned value. |

**Return value:** String. Returns all found messages as a tab-separated multi-line string. Each line consists of message time, message ID, message severity (either ERROR or WARNING), message source, user name, and message text separated by the tab character.

**Usage:** This statement can be used on Windows NT (NT/2000/XP/2003/VISTA/2008/7) systems only. You can also use the Windows NT System Event Log Viewer to read event log messages. You may want to use this statement to search Windows NT event logs for reported errors and this way effectively monitor other application that write to the event logs.

 **Note:**
The 24x7 Scheduler maintains its own event log that is different from the system event log. LogSearch does not search 24x7 Scheduler event log. However, if you configured 24x7 Scheduler to simultaneously log all messages to the Windows NT (NT/2000/XP/2003/VISTA/2008/7) System Event Log then LogSearch statement will also automatically search 24x7 Scheduler generated messages.

**See also:**
LogAddMessage

# LogSearchEx

**Description:** Search entries in the Windows NT (NT/2000/XP/2003/VISTA/2008/7) event log written by the specified application.

**Syntax:** LogSearchEx search_system, start_time, event_source, return

| Argument | Description |
|---|---|
| search_system | A boolean that indicates whether you want to search the system event log or the application event log. Specify TRUE for the system log or FALSE for the application event log. |
| start_time | A date-time value that is the earliest time of the messages that you want to search. Use this parameter as a “filter” for the event log. 24x7 Scheduler will search only messages that were logged since the start_time. |
| event_source | A string whose value is the name of the event source |

| | |
|---|---|
| | (e.g. name of the application that wrote the message). |
| Return | A string variable that receives the returned value. |

**Return value:** String. Returns all found messages as a tab-separated multi-line string. Each line consists of message time, message ID, message severity (either INFO, ERROR or WARNING), message source, user name, and message text separated by the tab character.

**Usage:** This statement can be used on Windows NT (NT/2000/XP/2003/VISTA/2008/7) systems only. You can also use the Windows NT System Event Log Viewer to read event log messages. You may want to use this statement to search Windows NT event logs for reported errors and this way effectively monitor other application that write to the event logs.

**Note:**
The 24x7 Scheduler maintains its own event log that is different from the system event log. LogSearchEx does not search 24x7 Scheduler event log. However, if you configured 24x7 Scheduler to simultaneously log all messages to the Windows NT System Event Log then LogSearchEx statement will also automatically search 24x7 Scheduler generated messages.

**See also:**
LogAddMessage
LogSearch
LogWaitForUpdate

# LogWaitForUpdate

**Description:** Suspends job execution and then enters an efficient wait state until either new event is written to the Windows NT (NT/2000/XP/2003/VISTA/2008/7) event log or the statement times out.

**Syntax:** LogWaitForUpdate wait_system, event_source, timeout, return_data, return_status

| Argument | Description |
|---|---|
| wait_system | A boolean that indicates whether you want to wait for the system event log or the application event log. Specify TRUE for the system log or FALSE for the application event log. |
| event_source | A string whose value is the name of the event source (e.g. name of the application that wrote the message). If you specify an empty string "", 24x7 Scheduler will resume after any new event record is added to the event log, otherwise it will resume only after a record is added by the specified event_source. |
| timeout | A number whose value is the maximum time interval within which you expect a new event record to be written to the specified event log. Use 0 timeout to allow infinite waiting. |
| return_data | A string variable that receives the returned event record data. |
| return_status | A boolean variable that receives the returned status value. |

**Return value:** String and Boolean. If the LogWaitForUpdate statement succeeds, the return_status is True and the return_data value contains new event log record as a tab-separated string, which consists of message time, message ID, message severity (either INFO, ERROR or WARNING), message source, user name, and message text. If the LogWaitForUpdate statement fails, the return_status value is False and return_data is empty string "".

**Usage:** This statement can be used on Windows NT (NT/2000/XP/2003/VISTA/2008/7) systems only. You can also use the Windows NT System Event Log Viewer to read event log messages. You may want to use this statement to monitor Windows NT event logs for new messages.

 **Note:**
The 24x7 Scheduler maintains its own event log that is different from the system event log. LogWaitForUpdate does not monitor 24x7 Scheduler event log. However, if you configured 24x7 Scheduler to simultaneously log all messages to the Windows NT (NT/2000/XP/2003/VISTA/2008/7) System Event Log then LogWaitForUpdate statement will also automatically monitor 24x7 Scheduler generated messages.

**See also:**

- LogAddMessage
- LogSearch
- LogSearchEx

# Miscellanious statements

## AgentTest

**Description:** Tests availability of the specified 24x7 Remote Agent.

**Syntax:** AgentTest agent, return

| Argument | Description |
|---|---|
| Agent | A string whose value is the name of the Remote Agent as it is specified in the Remote Agent profile. |
| Return | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE if the specified Agent is running and successful connection can be established between the main scheduler and this Remote Agent, otherwise returns FALSE.

**Usage:** Use this statement to check availability of the Remote Agent before executing JobRemoteRun statement or JobRun statement for a job that should be executed remotely by that Agent. You can also use that statement to check whether the remote computer hosting 24x7 Remote Agent is running.

**See also:**

- JobRun
- JobRemoteRun
- Remote Agents

# Call

**Description:** Calls a function within a dynamic-link library or an executable file.

**Syntax:** Call ( module, function, parameter-specification, has-result, [arguments], [result] )

| Argument | Description |
|---|---|
| Module | A string whose value is the filename of the DLL or EXE. |
| Function | A string whose value is the name of the function to run in the designated module. |
| Parameter-specification | A string whose value is indicating the formats of parameters passed to the function. Characters in the string represent C parameter types: Characters in upper case indicate that the corresponding parameters must be passed by reference, characters in lower case indicate that the corresponding parameters must be passed by value. Parameters of string and double data types are always passed by reference.<br>**Character Description**<br>U or u unsigned integer<br>L or L long<br>N or n short<br>C or c byte or char<br>F or f float<br>D or d double (always by ref)<br>S or s string (always by ref) |
| Has-result | A boolean whose value is indicating whether the calling function returns a result. |
| Parameters (optional) | A list of parameters that must be passed to the function. Order, number, and data types of parameters must match parameter-specification |
| result (optional) | A variable that receives the returned value. You must specify data type of the result value as the last character in the parameter-specification. |

**Return value:** See parameter description above.

**Usage:** It is recommended that you include the .DLL filename extension when specifying the name of the DLL. Without the extension, 24x7 Scheduler first looks through all the system search paths directories trying to find the file exactly as specified. Only after this fails does it add the .DLL extension and look through all the search paths again.

24x7 Scheduler supports only functions and subroutines in 32-bit DLLs and EXEs that use the _stdcall calling convention.
See examples for more information on usage.

**See also:**

String

Format

# ConsoleRead

**Description:** This statement can be used in standalone JAL scripts only to read user input from the command console.

**Syntax:** ConsoleRead( result )

| Argument | Description |
|---|---|
| Result | A string variable that receives the returned value. |

**Return value:** See parameter description above.

**Usage:** Use this statement in interactive processes to read data from the command console input buffer and remove it from the buffer. Note that this statement is blocking and will not return until user enters a value and presses Enter key.

**See also:**
ConsoleWrite
FileRead

# ConsoleWrite

**Description:** This statement can be used in standalone JAL scripts only, It can be used to write messages to the command console.

**Syntax:** ConsoleWrite( message )

| Argument | Description |
|---|---|
| Message | A string value that is output to the command console. |

**Return value:** See parameter description above.

**Usage:** Use this statement in standalone JAL scripts to write messages to the command console. For example, it can be used to write input prompts before calling ConsoleRead statement.

**See also:**
ConsoleRead
FileWrite

# DiskGetFreeSpace

**Description:** Retrieves the amount of free space on the specified disk.

**Syntax:** DiskGetFreeSpace path, freespace

| Argument | Description |
|---|---|
| Path | A string that is the disk letter you want to query or a file name or file path. In last two cases DiskGetFreeSpace uses the first character only. |
| Freespace | A numeric variable that receives the returned value |

**Usage:** The freespace receives the amount of free space in bytes.

**Important notes:**

- **Windows 95 OSR 1:** The DiskGetFreeSpace statement returns incorrect value for volumes that are larger than 2 gigabytes. That is because the operating system manipulates the disk properties so that computations with them yield the incorrect volume size.
- **Windows 95 OSR 2, Windows 98, Windows Me, Windows NT, Windows 2000 and Windows XP**: The DiskGetFreeSpaceEx statement is available on Windows 95 systems beginning with OEM Service Release 2 (OSR 2), Windows 98/Me/NT/2000/XP/2003/VISTA/2008/7. The GetDiskFreeSpaceEx statement returns correct values for all volumes, including those that are greater than 2 gigabytes.

**See also:**

DiskGetFreeSpaceEx

MemoryGetFree

# DiskGetFreeSpaceEx

**Description:** Retrieves the amount of free space on the specified disk.

**Syntax:** DiskGetFreeSpaceEx path, freespace

| Argument | Description |
|---|---|
| path | A string that is the disk letter you want to query or a file name or file path. In last two cases DiskGetFreeSpaceex uses the first character only. |
| freespace | A numeric variable that receives the returned value |

**Usage:** The freespace receives the amount of free space in bytes.

**Important notes:**

- **Windows 95 OSR1:** The DiskGetFreeSpaceEx statement is available on Windows 95 systems beginning with OEM Service Release 2 (OSR 2) only.
  The DiskGetFreeSpace statement returns incorrect value for volumes that are larger than 2 gigabytes. That is because the operating system manipulates the disk properties so that computations with them yield the incorrect volume size.
- **Windows 95 OSR 2, Windows 98/Me/NT/2000/XP/2003/VISTA/2008/7**: The DiskGetFreeSpaceEx statement is available on Windows 95 systems beginning with OEM Service Release 2 (OSR 2), Windows 98/Me/NT/2000/XP/2003/VISTA/2008/7. The GetDiskFreeSpaceEx statement returns correct value for all volumes, including those that are greater than 2 gigabytes.

**See also:**

DiskGetFreeSpace

MemoryGetFree

# InputBox

**Description:** Displays prompts in a dialog box, waits for the user to input values, and returns a string containing the contents of the dialog box.

**Syntax:** InputBox prompt_list, edit_style_list, return_values_list

| Argument | Description |
|---|---|
| prompt_list | A string whose value is the comma-separated list of prompts.<br>Each prompt is displayed on a separate row. Each row containing a prompt can display one or more lines of text. Lines are automatically wrapped to fit the prompt area of the input box. To force line breaks in a prompt, separate the lines using carriage return character (Char(13)), or carriage return–linefeed character combination (Char(13) & Char(10)) between each line. You can use **special ASCII characters** for this purpose |
| edit_style_list | A string whose value is the comma-separated list of required edit styles, one item for each prompt. Number of items in this value list must match number of items in the promt_list. The following edit styles are supported:<br>• EDIT<br>• YES/NO<br>• FILE BROWSE<br>• DIR BROWSE<br>• PROCESS BROWSE<br>• FTP BROWSE<br>• SFTP BROWSE<br>• MAIL PROFILE LIST<br>• REMOTE FILE BROWSE<br>• REMOTE DIR BROWSE<br>• REMOTE AGENT LIST<br>• DB PROFILE LIST<br>• QUEUE LIST |

| | |
|---|---|
| | • JOB BROWSE<br>• JOB FOLDER LIST<br><br>Multiple edit styles can be selected for a single InputBox dialog. Edit styles are the same as these used by the **job templates** and the **Job Wizard** |
| return_values_list | A string variable that receives the returned comma-separated list of user-entered values. |

**Return value:** A comma-separated string containing user-entered values. If only one prompt is specified, the returned string will contain only one value.

**Usage:** Use InputBox statement to display input dialogs that look and feel like input dialogs displayed by the job templates that you use for creating new jobs.

The names of edit styles are self-explaining.
Here is a brief description of what each style does:

| | |
|---|---|
| EDIT | Allows input of any text up-to 32000 characters long |
| YES/NO | Allows input of YES or NO values only |
| FILE BROWSE | Displays the standard file browse dialog box and allows selecting name of an existing file, the complete file name including path is returned |
| DIR BROWSE | Displays the standard folder browse dialog box and allows selecting name of an existing folder, the complete folder name including path is returned |
| PROCESS BROWSE | Displays process browse dialog and allows selecting name of the currently running process |
| FTP BROWSE | Displays FTP file browse dialog that looks and feels like the standard file browse dialog box, but allows selecting name of an existing file or directory on the remote FTP server, the complete file name including path is returned |
| SFTP BROWSE | Displays Secure FTP file browse dialog that looks and feels like the standard file browse dialog box, but allows selecting name of an existing file or directory on the remote Secure FTP server, the complete file name including path is returned |
| MAIL PROFILE LIST | Displays MAPI Profile dialog and allows selecting name of an existing mail profile |
| REMOTE FILE BROWSE | Displays file browse dialog that looks and feels like the standard file browse dialog box, but allows selecting name of an existing file on the remote computer running 24x7 Remote Agent or 24x7 Master Scheduler, the complete file name including path is returned |
| REMOTE DIR BROWSE | Displays folder browse dialog that looks and feels like the standard folder browse dialog box, but allows selecting name of an existing folder on the remote computer running 24x7 Remote Agent or 24x7 Master Scheduler, the complete folder name including path is returned |
| REMOTE AGENT LIST | Displays Remote Agent dialog and allows selecting name of an existing 24x7 Remote Agent profile |
| DB PROFILE LIST | Displays Database Profile dialog and allows selecting name of an existing database profile |
| QUEUE LIST | Displays Job Queue dialog and allows selecting name of an existing job queue |
| JOB BROWSE | Displays Job Tree dialog and allows selecting name of an existing job |
| JOB FOLDER BROWSE | Displays Job Folder dialog and allows selecting name of an existing job folder |

With the exception of YES/NO and EDIT styles, to enter the required values you can either use the input assistant or directly type or paste them from the clipboard. To paste a value from the Clipboard, use CTRL+V shortcut. To use the assistant, click the Build  button. The assistant will display a dialog with the specified edit style.

**Tips:**

- If the user input contains commas as a part of the returned value, you will not be able to properly parse the returned values from a multi-prompt dialog. The workaround is calling InputBox for each input value separately, in other words, for every input value you should display one dialog with one prompt only.
- It is a good idea to use GetToken statement to parse values returned from a multi-prompt InputBox

# MacroPlayBack

**Description:** Plays back GUI automation macros. GUI automation macros can be recorded using **MacroRecorder** - the GUI Automation utility, or be built dynamically in a JAL script job.

**Syntax:** MacroPlayBack macro, timeout, speed_ratio, show_control

| Argument | Description |
|---|---|
| macro | A string whose value is the macro-script that you want play. |
| timeout | A number whose value is playback timeout specified in seconds. If the playback does not finish in the specified time interval (e.g. timeout occurs) the 24x7 Scheduler aborts the playback. A 0 value disables timeout and allows 24x7 Scheduler to playback the macro to infinity. |
| speed_ratio | A number whose value is playback speed ratio as compared to the original speed of recorded events. Number 0 or 1 for the speed_ratio indicate that playback will run with the original speed. Number 2 indicates that playback will be twice as fast as compared to original speed, Number 3 indicates triple speed, and so on. |
| show_control | A boolean whose value indicates whether you want to see the playback control displayed in the top-right corner of the screen while the playback is in progress. The playback control might be helpful in job debugging and troubleshooting. Specify TRUE value if you wan the control to be displayed, and FALSE otherwise. |

**Return value:** None.

**Usage:** Use MacroPlayBack statement to automate various GUI applications that cannot be automated using conventional methods.

The macro for playback can be loaded from a file or created in a script.

**See also:**

MacroRecorder - GUI Automation utility

# MemoryGetFree

**Description:** Retrieves the amount of available free virtual memory.

**Syntax:** MemoryGetFree freebytes

| Argument | Description |
|---|---|
| freebytes | A numeric variable that receives the returned value |

**Usage:** The freebytes receives the amount of free virtual memory. This value includes amount of free space in the Windows swap file plus the amount of available free physical memory. As opposite to the MemoryGetFree statement the MemoryGetFreeEx statement returns the amount of free space in the Windows swap file only.

Divide the returned value by 1024 to calculate free memory in Kbytes. Divide the returned value by 1048576 to calculate free memory in Mbytes.

 **Note:**
24x7 Scheduler allocates some additional memory while executing JAL script. After execution, this memory is released back to the Operation System. Therefore, amount of free memory reported by the MemoryGetFree statement is usually less than amount of free memory available after the script run.

**See also:**
DiskGetFreeSpace
MemoryGetFreeEx

# MemoryGetFreeEx

**Description:** Retrieves the amount of available free virtual memory.

**Syntax:** MemoryGetFreeEx freebytes

| Argument | Description |
|---|---|
| freebytes | A numeric variable that receives the returned value |

**Usage:** The freebytes receives the amount of free virtual memory. This value includes amount of free space in the Windows swap file. As opposite to the MemoryGetFreeEx statement the MemoryGetFree statement returns the amount of free space in the Windows swap plus the amount of available free physical memory.

Divide the returned value by 1024 to calculate free memory in Kbytes. Divide the returned value by 1048576 to calculate free memory in Mbytes.

**Note:**
24x7 Scheduler allocates some additional memory while executing JAL script. After execution, this memory is released back to the Operation System. Therefore, amount of free memory reported by the MemoryGetFree statement is usually less than amount of free memory available after the script run.

**See also:**
DiskGetFreeSpace
MemoryGetFree

# MessageBox

**Description:** Displays a system Message Box with the text you specified, and YES/NO buttons.

**Syntax:** MessageBox message

| Argument | Description |
|---|---|
| message | A string that is the message you want to display |

**Usage:** The MessageBox displays standard Windows message box with the user-defined text and the prompt to press YES button to continue or press NO to stop the script. Pressing the NO button interrupts the script execution.

You should use this statement for Debugging purposes only. For instance, you can use it to display the value stored in a variable.

# Reboot

**Description:** Immediately logs off, shuts down, and restarts the system ("cold" reboot.).

**Syntax:** Reboot

**Usage**: This statement has no arguments.

**Note:**
During a shutdown operation, applications that are shut down are allowed a specific amount of time to respond to the shutdown request. If the time expires, Windows forces applications to close.

# ScreenCapture

**Description:** Captures screen shot and saves it in a file.

**Syntax:** ScreenCapture file

| Argument | Description |
|---|---|
| file | A string whose value is the name of the destination bitmap file. |

**Return value:** None.

**Usage:** Use ScreenCapture statement to automate screen-capturing functions. ScreenCapture can be used together with the FileTransfer statement to automatically capturing screen shots on remote computers and then transferring saved bitmap files to the administrator's computer.

The size of the destination files depends on your screen resolution and number of colors selected for the display.

 **Important Note:** ScreenCapture fails if it is called when a screen saver is running.

**See also:**
WindowCapture

# VBScriptExecute

**Description:** Executes specified Visual Basic Script

**Syntax:** VBScriptExecute script

| Argument | Description |
|---|---|
| Script | A string whose value is the Visual Basic Script that you want to execute from your JAL script. |

**Return value:** None.

**Usage:** Use VBScriptExecute statement to execute methods available in VBScript and not available in JAL. For example, you can use it to add COM automation to your existing JAL jobs.

**See also:**
VBScript Language Reference
Scripting Conventions

# WindowCapture

**Description:** Captures part of the screen allocated by the specified window and saves it in a file.

**Syntax:** WindowCapture handle, file

| Argument | Description |
|---|---|
| handle | A number whose value is the handle of the window that you want to capture. |
| file | A string whose value is the name of the destination bitmap file. |

**Return value:** None.

**Usage:** Use WindowCapture statement to automate screen-capturing functions. WindowCapture can be used together with the other windows statements that can return the handle of the window.

The size of the destination files depends on your screen resolution and number of colors selected for the display.

 **Important Note:** WindowCapture fails if it is called when a screen saver is running.

**See also:**
ScreenCapture
WindowFind

# Yield

**Description:** Yields execution so that the operating system can process other events.

**Syntax:** Yield

**Usage**: This statement has no arguments.

# Numeric statements

## Add

**Description:** Obtains the result of an addition operation.

**Syntax:** Add n1, n2, sum

| Argument | Description |
|---|---|
| n1 | The number to which you want to add n2. |
| n2 | The number you want to add to n1 |
| sum | A numeric variable that receives the returned value |

**Return value:** Returns the sum of n1 and n2 so that sum **=** n1 + n2

**Usage:** You can also use keyword Plus instead of Add.

**See also:**

Subtract
Multiply
Divide
Mod

## Ceiling

**Description:** Determines the smallest whole number that is greater than or equal to a specified limit.

**Syntax:** Ceiling n, return

| Argument | Description |
|---|---|
| n | The number for which you want the smallest whole number that is greater than or equal to it |
| return | A numeric variable that receives the returned value |

**Return value:** Returns the smallest whole number that is greater than or equal to n.

**See also:**

Floor

# Divide

**Description:** Obtains the result of a division operation.

**Syntax:** Divide n1, n2, div

| Argument | Description |
|---|---|
| n1 | The number you want to divide by n2 |
| n2 | The number you want to divide into n1 |
| div | A numeric variable that receives the returned value |

**Return value:** Returns the result of division n1 by n2 so that div **=** n1 / n2

**See also:**

Subtract
Multiply
Add
Mod

# Floor

**Description:** Determines the largest whole number less than or equal to a number.

**Syntax:** Floor n, return

| Argument | Description |
|---|---|
| n | The number for which you want the largest whole number that is less than or equal to it |
| return | A numeric variable that receives the returned value |

**Return value:** Returns the largest whole number less than or equal to n.

**See also:**

Ceiling

# Max

**Description:** Determines the larger of two numbers.

**Syntax:** Max n1, n2, return

| Argument | Description |
|---|---|
| n1 | The number to which you want to compare n2 |
| n2 | The number to which you want to compare n1 |
| return | A numeric variable that receives the returned value |

**See also:**
Min

## Min

**Description:** Determines the smaller of two numbers.

**Syntax:** Min n1, n2, return

| Argument | Description |
|---|---|
| n1 | The number to which you want to compare n2 |
| n2 | The number to which you want to compare n1 |
| return | A numeric variable that receives the returned value |

**See also:**
Max

## Mod

**Description:** Obtains the remainder (modulus) of a division operation.

**Syntax:** Mod n1, n2, return

| Argument | Description |
|---|---|
| n1 | The number you want to divide by n2 |
| n2 | The number you want to divide into n1 |
| return | A numeric variable that receives the returned value |

**Return value:** Returns the reminder of a division using the following formula: return = n2 - (floor( n2 / n1) * n1)

**See also:**
Subtract

Multiply
Divide
Add

## Multiply

**Description:** Obtains the result of a multiplication operation.

**Syntax:** Multiply n1, n2, return

| Argument | Description |
|---|---|
| n1 | The number you want to multiply by n2 |
| n2 | The number you want to be a multiplier for n1 |
| return | A numeric variable that receives the returned value |

**Return value:** Returns the result of multiplication n1 by n2 so that return **=** n1 * n2.

**See also:**
Subtract
Add
Divide
Mod

## Power

**Description:** Obtains the result of an exponentiation operation.

**Syntax:** Power n1, n2, return

| Argument | Description |
|---|---|
| n1 | The number you want to be a base for n2 |
| n2 | The number you want to be a exponent for n1 |
| return | A numeric variable that receives the returned value |

**Return value:** Returns the result of exponentiation n1 by n2 so that return **=** $n1^{n2}$.

**See also:**
Subtract
Multiply
Divide
Mod
Add

# Round

**Description:** Rounds a number to the specified number of decimal places.

**Syntax:** Round n1, n2, return

| Argument | Description |
|---|---|
| n1 | The number you want to round |
| n2 | The number of decimal places to which you want to round n1. Valid values are 0 through 18 |
| return | A numeric variable that receives the returned value |

**Usage:** Returns the result of rounding n1 to n2 decimal places.

**See also:**

Floor
Ceiling
Divide

# Subtract

**Description:** Obtains the result of a subtraction operation.

**Syntax:** Subtract n1, n2, sub

| Argument | Description |
|---|---|
| n1 | The number to which you want to add n2. |
| n2 | The number you want to add to n1 |
| sub | A numeric variable that receives the returned value |

**Return value:** Returns the subtraction of n1 and n2 so that sub **=** n1 - n2

**Usage:** You can also use keyword Minus instead of Subtract.

**See also:**

Add
Multiply
Divide

Mod

# Print statements

## FilePrint

**Description:** Sends the specified file to the default printer

**Syntax:** FilePrint filename

| Argument | Description |
|---|---|
| filename | A string whose value is the name of the file you want to print. If filename is not on the operating system's search path, you must enter the fully qualified name |

**Return value:** None.

**See also:**

PrinterSetDefault

## Print

**Description:** Prints the specified string.

**Syntax:** Print s

| Argument | Description |
|---|---|
| s | The string you want to print. If the string includes carriage return / new line character pairs (CR/NL), the string will be printed on multiple lines |

**Return value:** None.

**See also:**

FilePrint

PrinterSetDefault

# PrinterGetDefault

**Description:** Reports name of the default printer.

**Syntax:** PrinterGetDefault s

| Argument | Description |
|---|---|
| s | A string variable that receives the name of the default printer |

**Return value:** None.

**Usage:** You can use PrinterGetDefault to retrieve name of the default printer before you change it using PrinterSetDefault so that you can restore the default printer after some processing done.

**See also:**

FilePrint

PrinterSetDefault

# PrinterPurgeAllJobs

**Description:** Discards all pending print jobs from the specified print queue.

**Syntax:** PrinterPuregAllJobs s

| Argument | Description |
|---|---|
| s | A string variable whose value is the name of a printer for which you want to discard all pending print jobs |

**Return value:** None.

**See also:**

PrinterGetDefault

PrinterSetDefault

# PrinterSetDefault

**Description:** Changes the default printer.

**Syntax:** PrinterSetDefault s

| Argument | Description |
|---|---|
| S | A string whose value is the name of the printer that you want to be a default printer |

**Return value:** None.

**See also:**

FilePrint
PrinterGetDefault

# Process statements

## IsTaskRunning

**Description:** Reports whether the specified task is running.

**Syntax:** IsTaskRunning module, return

| Argument | Description |
|---|---|
| module | A string whose value is the name of the main program module. Usually the module name matches name of the main program executable file. If the module extension is omitted, the default library extension (.DLL) is appended. The module string can include a trailing point character (.) to indicate that the module name has no extension. The string does not have to specify a path. The name is compared (case independently) to the names of modules currently running |
| return | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE when the specified module is running, and FALSE otherwise.

**Usage:** You can use the Windows Task Manager to find out the module name. For example the module name for MS Word is WINWORD.EXE. When you know the module name you can use IsTaskRunning statement to check state of the required application. For instance, in order to fax something, you may want to check presence of Delrina WinFax instance in a memory before establishing a DDE connection to the Delrina DDE server. If IsTaskRunning returns FALSE then you can call Run statement to open an instance of Delrina WinFax .
Alternatively, you can use ProcessList statement to get a string listing all processes and their process IDs.

**See also:**

Run
ProcessGetID
ProcessList

ProcessKill
ServiceGetStatus

# JobProcessID

**Description:** Obtains Windows process identifier for the current 24x7 job process. If the job is running attached, this is the ID of the main 24x7 process; otherwise it is the ID of the detached job process.

**Syntax:** JobProcessID pid

**Return value:** Number. System process ID for the process running the job in which the JobProcessID statement is being executed

**Usage:** Use JobProcessID in job control scripts. For example, detached jobs can use this ID to abort job execution using ProcessKill statement.

**Important note:**

- Process ID is assigned by the operation system. Process ID is only unique at the time of the job run. If the job is not setup to run detached, process ID is the same as ID of the main 24x7.exe process.

**See also:**

JobThreadID
ProcessGetID
ProcessKill

# JobThreadID

**Description:** Obtains Windows thread identifier for the current 24x7 job. If the job is running attached, this is the thread ID of the job process within 24x7 job queue; otherwise it is the main thread ID of the detached job process.

**Syntax:** JobThreadID tid

**Return value:** Number. System thread ID for the thread running the job in which the JobThreadID statement is being executed

**Usage:** Use JobThreadID in asynchronous and/or detached jobs to obtain unique job instance identifier. You can also use in advanced job control scenarios. For example, an asynchronous background job can return this ID to an external process management application. That application can use the obtained thread ID to suspend the job and put it to sleep. Later it can wake up the job and resume the execution from the point when the job execution has been previously suspended.

**Important note:**

- Thread ID is assigned by the operation system. Thread ID is unique in the system. Yet if the job is run synchronous and non-detached, the thread ID will be always the same for all job instances run by the same 24x7 Scheduler instance and will match thread ID of the associated job queue manager process.

**See also:**

JobProcessID
ProcessGetID
ProcessKill

# ProcessGetID

**Description:** Reports process ID for the specified module.

**Syntax:** ProcessGetID filename, return

| Argument | Description |
|---|---|
| filename | A string whose value is the full name of the process main program module. Usually the module name matches name of the main program executable file. If the module extension is omitted, the default library extension (.DLL) is appended. The module string can include a trailing point character (.) to indicate that the module name has no extension. The string has to include a file path to the desired module. The name is compared (case independently) to the names of modules currently running |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns Windows process identifier for the specified module. A process identifier can change for each run of the program for which you want to find process ID.

**See also:**

ProcessGetHandle
ProcessGetWindow
JobProcessID
ProcessKill
WindowGetProcess

# ProcessGetHandle

**Description:** Reports process handle for the specified module.

**Syntax:** ProcessGetHandle module, return

| Argument | Description |
|---|---|
| module | A string whose value is the full name of the process main program module. Usually the module name matches name of the main program executable file. If the module extension is omitted, the default library extension (.DLL) is appended. The module string can include a trailing point character (.) to indicate that the module name has no extension. The string has to include a file path to the desired module. The name is compared (case independently) to the names of |

| | |
|---|---|
| | modules currently running |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns process handle for the specified module. A process handle can change for each run of the program for which you want to find out process handle.

**Usage:** The returned handle is not global and should not be passed as a parameter to other Windows applications.

**See also:**

ProcessGetID
ProcessGetWindow
ProcessKill
WindowGetProcess

# ProcessKill

**Description:** Terminates the specified process and all of its threads.

**Syntax:** ProcessKill processid

| Argument | Description |
|---|---|
| processid | A number whose value is the id of the process which you want to terminate |

**Return value:** None.

**Usage:** The ProcessKill statement can be used to unconditionally cause a process to exit. Use it only in extreme circumstances. You can use ProcessGetID statement to find the process ID for the desired module (program). You can also use WindowGetProcess statement to get process ID for the desired window.

**Note:**
Termination of a process does not always forces termination of all its child processes!

**See also:**

ProcessGetID
WindowGetProcess
JobProcessID

# ProcessList

**Description:** Reports all running processes and their process IDs.

**Syntax:** ProcessList return

| Argument | Description |
|---|---|
| return | A string variable that receives the returned value |

**Return value:** String. Returns Ids and names of all running process as multi-line string. Each line consists of process ID following by the tab character and process name.

**Usage:** Use the returned value to find out process name for the desired application. The application must be running at the when you call ProcessList statement.

 **Note:** Because of security restrictions in Windows NT (NT/2000/XP/2003/VISTA/2008/7) the returned string may include not all running processes.

**See also:**

ProcessGetID
ProcessGetWindow
ProcessKill
WindowGetProcess

# ProcessGetWindow

**Description:** Finds and reports the handle of a first window whose process ID matches the specified process ID.

**Syntax:** ProcessGetWindow processID, return

| Argument | Description |
|---|---|
| processID | A number whose value is the ID of the process for which you want to find first window |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns the handle of the first window that Window's finds for the specified processID. A process ID can change for each run of the program for which you want to find the first window.

**Usage:** The returned handle is global and uniquely identifies the window instance. You can use ProcessGetID statement to find out the ID of desired module.

**See also:**

ProcessGetHandle
ProcessGetID
ProcessKill
WindowGetProcess

# Run

**Description:** Runs the specified program or document.

**Syntax:** Run command, dir, return

| Argument | Description |
|---|---|
| command | A string whose value is the full or partial path and filename of the module to execute.<br><br>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.<br><br>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:<br><br>1 The directory from which the 24x7 Scheduler loaded.<br>2 The current directory.<br>3 The 32-bit Windows system directory. The name of this directory is SYSTEM32.<br>4 The 16-bit Windows system directory. The name of this directory is SYSTEM.<br>5 The Windows directory. The name of this directory is WINDOWS or WINNT.<br>6 The directories that are listed in the PATH environment variable.<br><br>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.<br><br>Program name can be followed by **command line parameters**. You can use **macro-parameters** and **system environment variables** within program name and command line parameters string to pass dynamic information (such as the current month) to the scheduled program. |
| dir | A string whose value is 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 the 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. You can use **macro-parameters** and **system environment variables** within directory name. |

| | |
|---|---|
| return | A numeric variable that receives the ID of the created process. |

**Return value:** Number. Returns the ID of the created process. You can use that ID to control and manipulate further process execution.

**Usage:** You can use Run for any application that you can run from the operating system. If you specify command line parameters, the application determines the meaning of those parameters. A typical use for command line parameters is to identify a data file to be opened when the program is executed.
In order to run a document, its extension must be registered. For example, if you want to start a MDB file that has AutoExec macro, you must have MDB file extension registered as a MS Access database application.

**See also:**

RunConfig
RunAndWait
RunAsUser
RunWithInput
SendKeys
Wait

# RunAsUser

**Description:** Runs the specified program or document using different Windows NT (NT/2000/XP/2003/VISTA/2008/7) user logon.

**Syntax:** RunAsUser command, dir, user, password, domain, return

| Argument | Description |
|---|---|
| command | A string whose value is the full or partial path and filename of the module to execute.<br><br>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.<br><br>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:<br><br>7 The directory from which the 24x7 Scheduler loaded.<br>8 The current directory.<br>9 The 32-bit Windows system directory. The name of this directory is SYSTEM32.<br>10 The 16-bit Windows system directory. The name of this directory is SYSTEM. |

| | |
|---|---|
| | 11 The Windows directory. The name of this directory is WINDOWS or WINNT.<br>12 The directories that are listed in the PATH environment variable.<br><br>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.<br><br>Program name can be followed by **command line parameters**. You can use **macro-parameters** and **system environment variables** within program name and command line parameters string to pass dynamic information (such as the current month) to the scheduled program. |
| dir | A string whose value is 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 the 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. You can use **macro-parameters** and **system environment variables** within directory name. |
| user | A string variable that specifies the user name. This is the name of the user account to log on to. |
| password | A string variable that specifies the password for the user account specified by user. |
| domain | A string variable that specifies the domain or server to log on to. If this parameter is ".", Windows NT (NT/2000/XP/2003/VISTA/2008/7) searches only the local account database for the account specified in user. |
| return | A numeric variable that receives the ID of the created process. |

**Return value:** Number. Returns the ID of the created process. You can use that ID to control and manipulate further process execution.

**Usage:** You can use RunAsUser for any application that you can run from the operating system. If you specify command line parameters, the application determines the meaning of those parameters. A typical use for command line parameters is to identify a data file to be opened when the program is executed.
In order to run a document, its extension must be registered. For example, if you want to start a MDB file that has AutoExec macro, you must have MDB file extension registered as a MS Access database application.

**Important Notes**:

- RunAsUser statement is **not supported on Windows 95/98/Me** systems.
- Windows NT (NT/2000/XP/2003/VISTA/2008/7) considers the user as logged on until the process (and all child processes) have ended.
- The process created by the RunAsUser statement is non-interactive, that is, it runs on a desktop that is not visible and cannot receive user input. Also, the process inherits the environment of the 24x7 Scheduler, rather than the environment associated with the specified user.

**See also:**

Run
RunAsUserAndWait
RunConfig
ProcessKill

# RunAndWait

**Description:** Starts specified program or document and enters an efficient wait state until this process finishes or until the timeout interval elapses. In the last case the 24x7 Scheduler forcedly terminates the process.

**Syntax:** RunAndWait command, dir, timeout, return

| Argument | Description |
|---|---|
| Command | A string whose value is the full or partial path and filename of the module to execute.<br><br>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.<br><br>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:<br><br>1 The directory from which the 24x7 Scheduler loaded.<br>2 The current directory.<br>3 The 32-bit Windows system directory. The name of this directory is SYSTEM32.<br>4 The 16-bit Windows system directory. The name of this directory is SYSTEM.<br>5 The Windows directory. The name of this directory is WINDOWS or WINNT.<br>6 The directories that are listed in the PATH environment variable.<br><br>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.<br><br>Program name can be followed by **command line** |

| | |
|---|---|
| | **parameters**. . You can use **macro-parameters** and **system environment variables** within program name and command line parameters string to pass dynamic information (such as the current month) to the scheduled program. |
| Dir | A string whose value is 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 the 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. . You can use **macro-parameters** and **system environment variables** within directory name. |
| Timeout | A number whose value is the maximum time interval within which you allow the specified program to run. Use 0 timeout to allow infinite waiting. |
| Return | A numeric that receives the ID of the created process. |

**Return value:** Number. Returns the ID of the created process.

**Usage:** You can use RunAndWait for any application that you can run from the operating system. If you specify command line parameters, the application determines the meaning of those parameters. A typical use for command line parameters is to identify a data file to be opened when the program is executed.
In order to run a document, its extension must be registered. So that if you want to start MDB files that has AutoExec macro you must have MDB file extension registered as a MS Access database application.
Use RunAndWait when you have a complex job with one or more program dependencies so the next program starts upon completion of the process specified by command. For example, you can call RunAndWait first to execute a program that creates huge data file. Then you call FileZip statement that compresses the produced data file. After that, you can use MailSendWithAttachment statement to email the archive created by the FileZip.

**See also:**

Run
RunAsUserAndWait
RunWithInput
RunConfig
ProcessGetExitCode
SendKeys
Wait

# RunAsUserAndWait

**Description:** Starts specified program or document and enters an efficient wait state until this process finishes or the timeout interval elapses. In the latter case, the 24x7 Scheduler forcedly terminates the process. The process is started using different Windows NT (NT/2000/XP/2003/VISTA/2008/7) user logon.

**Syntax:** RunAsUserAndWait command, dir, timeout, user, password, domain, return

| Argument | Description |
|---|---|
| command | A string whose value is the full or partial path and filename of the module to execute.<br><br>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.<br><br>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:<br><br>1 The directory from which the 24x7 Scheduler loaded.<br>2 The current directory.<br>3 The 32-bit Windows system directory. The name of this directory is SYSTEM32.<br>4 The 16-bit Windows system directory. The name of this directory is SYSTEM.<br>5 The Windows directory. The name of this directory is WINDOWS or WINNT.<br>6 The directories that are listed in the PATH environment variable.<br><br>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.<br><br>Program name can be followed by **command line parameters**. . You can use **macro-parameters** and **system environment variables** within program name and command line parameters string to pass dynamic information (such as the current month) to the scheduled program. |
| dir | A string whose value is 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 the 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. . You can use **macro-parameters** and **system environment variables** within directory name. |
| timeout | A number whose value is the maximum time interval within which you allow the specified program to run. Use 0 timeout to allow infinite waiting. |
| user | A string variable that specifies the user name. This is the name of the user account to log on to. |
| password | A string variable that specifies the password for the user account specified by user. |

| | |
|---|---|
| domain | A string variable that specifies the domain or server to log on to. If this parameter is ".", Windows NT (NT/2000/XP/2003/VISTA/2008/7) searches only the local account database for the account specified in user. |
| return | A numeric variable that receives the ID of the created process. |

**Return value:** Number. Returns the ID of the created process.

**Usage:** You can use RunAsUserAndWait for any application that you can run from the operating system. If you specify command line parameters, the application determines the meaning of those parameters. A typical use for command line parameters is to identify a data file to be opened when the program is executed.
In order to run a document, its extension must be registered. So that if you want to start MDB files that has AutoExec macro you must have MDB file extension registered as a MS Access database application.
Use RunAsUserAndWait when you have a complex job with one or more program dependencies so the next program starts upon completion of the process specified by command. For example, you can call RunAsUserAndWait first to execute a program that creates huge data file. Then you call FileZip statement that compresses the produced data file. After that, you can use MailSendWithAttachment statement to email the archive created by the FileZip.

**Important Notes**:

- RunAsUserAndWait statement is **not supported on Windows 95/98/Me** systems.
- Windows NT (NT/2000/XP/2003/VISTA/2008/7) considers the user as logged on until the process (and all child processes) have ended.
- The process created by the RunAsUserAndWait statement is non-interactive, that is, it runs on a desktop that is not visible and cannot receive user input. Also, the process inherits the environment of the 24x7 Scheduler, rather than the environment associated with the specified user.

**See also:**

Run
RunAsUser
RunAndWait
RunConfig
ProcessGetExitCode
Wait

# RunWithInput

**Description:** Starts the specified program or document, sends one or more keystrokes to the active window as if typed at the keyboard, and then enters an efficient wait state until the started process finishes or until the timeout interval elapses. In the last case, the 24x7 Scheduler forcedly terminates the process.

**Syntax:** RunWithInput command, dir, init_time, keystroke, timeout, return

| Argument | Description |
|---|---|
| command | A string whose value is the full or partial path and filename of the module to execute.<br>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.<br><br>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:<br><br>1 The directory from which the 24x7 Scheduler loaded.<br>2 The current directory.<br>3 The 32-bit Windows system directory. The name of this directory is SYSTEM32.<br>4 The 16-bit Windows system directory. The name of this directory is SYSTEM.<br>5 The Windows directory. The name of this directory is WINDOWS or WINNT.<br>6 The directories that are listed in the PATH environment variable.<br><br>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.<br><br>Program name can be followed by **command line parameters**. . You can use **macro-parameters** and **system environment variables** within program name and command line parameters string to pass dynamic information (such as the current month) to the scheduled program. |
| dir | A string whose value is 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 the 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. You can use **macro-parameters** and **system environment variables** within directory name. |
| init_time | A number whose value is the time in seconds that you want the 24x7 to wait before sending a keystroke to the spawned program, allowing the program to start and initialize. |
| keystroke | A string whose value is the keystroke you want to send to the started program. |
| timeout | A number whose value is the maximum time interval within which you want the specified program to finish. Use 0 timeout to allow infinite waiting. |
| return | A numeric that receives the ID of the created process. |

**Return value:** Number.

**Usage:** You can emulate any keystrokes that you might need to further automate the actions of this program. You can use RunWithInput for any application that you can run from the operating system. If you specify command line parameters, the application determines the meaning of those parameters. A typical use for command line parameters is to identify a data file to be opened when the program is executed.
In order to run a document, its extension must be registered. So that if you want to start MDB files that has AutoExec macro you must have MDB file extension registered as a MS Access database application.

Use RunWithInput when you have a complex job with one or more program dependencies so the next program starts upon completion of the process specified by command. For instance, you can call RunWithInput first to execute a program that downloads data from database. You send a keystroke to emulate an operator login to database and the command to save some data in an external file. Then you can call RunAndWait statement to execute a program that compresses the produced data file. After that, you can use MailSendWithAttachment statement to email the archive created before.

**See also:**

Run
RunAndWait
RunConfig
ProcessGetExitCode
SendKeys
Wait

# RunConfig

**Description:** Configures run-time options for subsequent Run… statements. The new options affect only the job in which the RunConfig statement is executed.

**Syntax:** RunConfig option, new_value

| Argument | Description |
|---|---|
| option | A string whose value is the option name. The following options are supported:<br>• WINDOW<br>• DESKTOP<br>• TITLE<br>• CPU |
| new_value | A string or number whose value is the new option value. |

**Return value:** None.

**Usage:** You can use RunConfig statement to configure run-time options for the following **Run**, **RunAndWait** and all other Run… statements.

- The WINDOW option controls size and state of the graphical windows that will be allocated by the system for started processes. For the new_value you can specify either of the following values: "HIDE", "MINIMIZE", "MAXIMIZE", "NORMAL".

- The DESKTOP option is supported only on Windows NT based systems (Windows NT 4, Windows 2000, Windows XP, Windows NET and better). You can use this option to specify on which virtual Desktop you want to start the processes. Using this option in 24x7 Scheduler NT service you can for example start a graphical process running on the user Desktop from. In fact the default graphical user desktop name is

winsta0\default. For more information about Desktops and their names see Microsoft Windows SDK documentation.
Example:

```
RunConfig( "DESKTOP", "winsta0\default" )
// run process
Dim( process_id, number )
Run( "myprocess.exe", "", process_id )
```

- The TITLE option can be used to assign custom title to DOS console process windows. This can be used for user convinience so the user can easily identify different processes. It can be also with the **WindowFind** statement, which can be called later to find the previously created console window. You can use the following code as a template:

```
// set unique title
Dim( unique_tittle, string )
Timer( 0, unique_tittle )
RunConfig( "TITLE", unique_title )
// run process
Dim( process_id, number )
Run( "myprocess.exe", "", process_id )
... some other processing is performed here ...
// Find and close the created process window
Dim( window_handle, number )
WindowFind( unique_title, window_handle )
WindowClose( window_handle )
```

- The CPU option is supported only on Windows NT based systems (Windows NT 4, Windows 2000, Windows XP, Windows NET and better). You can use this option to bind different processes to different CPUs. Obviously this option is useful only on multi-processor systems. By distributing load across multiple CPUs you can achieve better system performance and allow run of more concurrent jobs. Please keep in mind the 24x7 Scheduler job engine always runs on the first CPU. For new_value parameter you should specify the desired CPU number such as 1, 2, 3 and so on. Specify 0 value resets the CPU option and allows the system to decide which CPU to use for the processes.

**See also:**
Run
RunAndWait
RunAsUser
RunAsUserAndWait
RunWithInput

# ProcessGetExitCode

**Description:** Obtains exit code of the last process that was ran from the script using RunAndWait or RunWithInput statements.

**Syntax:** ProcessGetExitCode return

| Argument | Description |
|---|---|
| return | A numeric variable that receives the exit code of the last process. |

**Return value:** Number. Returns the exit code the last process.

**Usage:** ProcessGetExitCode returns either of the following values:
-1 -- if an error occurs (such as program not found or cannot be started)
or
-100 -- if the process was terminated by 24x7 Scheduler because it timed out
or
process exit code.

**See also:**

RunAndWait
RunWithInput
WindowFind
ProcessGetWindow

# SendKeys

**Description:** Sends one or more keystrokes to the active window as if typed at the keyboard.

**Syntax:** SendKeys keystroke

| Argument | Description |
|---|---|
| keystroke | A string whose value is the keystrokes you want to send |

**Return value:** None.

**Usage:** Each key is represented by one or more characters. To specify a single keyboard character, use the character itself. For example, to represent the letter A, use "A" for string. To represent more than one character, append each additional character to the one preceding it. To represent the letters A, B, and C, use "ABC" for string. The left and right parentheses ( ) have special meanings to SendKeys. To specify one of these characters, enclose it within braces ({}). For example, to specify the plus sign, use {(}. To specify brace characters, use {{} and {}}.

To specify characters that aren't displayed when you press a key, such as ENTER or TAB, and keys that represent actions rather than characters, use the codes shown below:

| Key | Code |
|---|---|
| BACKSPACE | {BACKSPACE}, {BS}, or {BKSP} |
| BREAK | {BREAK} |
| CAPS LOCK | {CAPSLOCK}, {CAPSLOCK}, or {CAP} |
| DEL or DELETE | {DELETE} or {DEL} |
| DOWN ARROW | {DOWN} |
| END | {END} |
| ENTER | {ENTER} or {CR}, {RETURN} |
| ESC | {ESC} or {ESCAPE} |
| HOME | {HOME} |
| INS or INSERT | {INSERT} or {INS} |

| | |
|---|---|
| LEFT ARROW | {LEFT} |
| NUM LOCK | {NUMLOCK} or {NUM LOCK} |
| PAGE DOWN | {PGDN}, {PAGE DOWN}, {PAGEDOWN} |
| PAGE UP | {PGUP}, {PAGE UP}, or {PAGE UP} |
| PRINT SCREEN | {PRTSC}, {PRINT}, or {PRINT SCREEN} |
| RIGHT ARROW | {RIGHT} |
| SCROLL LOCK | {SCROLLLOCK} or {SCROLL LOCK} |
| TAB | {TAB} |
| UP ARROW | {UP} |
| F1 | {F1} |
| F2 | {F2} |
| F3 | {F3} |
| F4 | {F4} |
| F5 | {F5} |
| F6 | {F6} |
| F7 | {F7} |
| F8 | {F8} |
| F9 | {F9} |
| F10 | {F10} |
| F11 | {F11} |
| F12 | {F12} |
| Numpad 0 | {NUMPAD0} or {NUMPAD 0} |
| Numpad 1 | {NUMPAD1} or {NUMPAD 1} |
| Numpad 2 | {NUMPAD2} or {NUMPAD 2} |
| Numpad 3 | {NUMPAD3} or {NUMPAD 3} |
| Numpad 4 | {NUMPAD4} or {NUMPAD 4} |
| Numpad 5 | {NUMPAD5} or {NUMPAD 5} |
| Numpad 6 | {NUMPAD6} or {NUMPAD 6} |
| Numpad 7 | {NUMPAD7} or {NUMPAD 7} |
| Numpad 8 | {NUMPAD8} or {NUMPAD 8} |
| Numpad 9 | {NUMPAD9} or {NUMPAD 9} |
| Numpad * | {NUMPAD*} or {NUMPAD *} |
| Numpad + | {NUMPAD+} or {NUMPAD +} |
| Numpad - | {NUMPAD-} or {NUMPAD -} |
| Numpad / | {NUMPAD/} or {NUMPAD /} |
| Numpad . | {NUMPAD.} or {NUMPAD .} |
| Numpad ENTER | {NUMPADENTER}, {NUMPAD ENTER}, {NUMPAD CR}, or {NUMPAD RETURN} |

To specify keys combined with any combination of the SHIFT, CTRL, and ALT keys, precede the key code with one or more of the following codes:

| Key | Code |
|---|---|

| | |
|---|---|
| SHIFT | {SHIFT} |
| CTRL | {CTRL} |
| ALT | {ALT} |

To specify that any combination of SHIFT, CTRL, and ALT should be held down while several other keys are pressed, enclose the code for those keys in parentheses. For example, to specify to hold down SHIFT while E and C are pressed, use "{SHIFT}(EC)". To specify to hold down SHIFT while E is pressed, followed by C without SHIFT, use "{SHIFT}EC" .
To specify that a delay is needed between sending several keys, use the {WAIT} code. This will put the 24x7 Scheduler in a 1 second sleep mode. To specify longer delay, use the form {WAIT number}. You must put a space between WAIT and number. For example, {WAIT 5} means sleep for 5 seconds.

**Note:**
You can't use SendKeys to send keystrokes to an application that is not designed to run in Microsoft Windows.

**See also:**

RunWithInput
WindowActivate
Wait

# Wait

**Description:** Enters an efficient wait state until the waitinterval elapses.

**Syntax:** Wait waitinterval

| Argument | Description |
|---|---|
| Waitinterval | A number whose value is the time interval expressed in seconds during which you want the 24x7 Scheduler to “sleep". |

**Return value:** None.

**Usage:** You can use Wait to temporarily pause script execution. For example you may want to pause the script for a few second to allow other processing to finish before continuing the script.

**See also:**

Run
RunAndWait

# WindowGetProcess

**Description:** Obtains process ID for the specified window.

**Syntax:** WindowGetProcess handle, return

| Argument | Description |
|---|---|
| handle | A number whose value is the handle of the window for which you want to find its process ID |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Return process ID for the specified window.

**Usage:** Use WindowGetProcess to pass returned value to the ProcessKill statement when you cannot gracefully close that process and want to terminate the process anyway.

**See also:**

ProcessGetWindow
ProcessKill
ProcessGetID
WindowClose
WindowFind
WindowGetActive

# RAS statements (RAS for Windows platforms)

## RASDial

**Description:** Establishes a RAS (remote access service) connection between a RAS client and a RAS server.

**Syntax:** RASDial phonebook_entry, user_name, password, return

| Argument | Description |
|---|---|
| Phonebook_entry | A string whose value is the phonebook entry to use to establish the connection. |
| user_name | A string whose value is the user's user name. This string is used to authenticate the user's access to the remote access server. |
| Password | A string whose value is the user's password. This string is used to authenticate the user's access to the remote access server. |
| Return | A numeric variable that receives the returned value. |

**Return value:** Number. The returned value is the RAS connection handle. Use the returned value as a parameter for other RAS statements.

**Usage:** 24x7 Scheduler executes RASDial statement synchronously meaning that the control is not returned and the next statement is not executed until RASDail has completed successfully or failed.

**See also:**

RASHangUp

RASGetStatus

# RASGetStatus

**Description:** Retrieves information on the current status of the specified RAS (remote access service) connection.

**Syntax:** RASGetStatus connection, return

| Argument | Description |
|---|---|
| Connection | A number whose value is the connection handle returned by RASDial statement. |
| Return | A string variable that receives the returned value. |

**Return value:** String. Returns information on the current status of the specified remote access connection. You can use this statement to determine the connection status before starting or closing other applications that use RAS connections. The returned status can be one of the following:

- Open Port
- Port Opened
- Connect Device
- Device Connected
- All Devices Connected
- Authenticate
- Auth. Notify
- Auth. Retry
- Auth. Callback
- Auth. Change Password
- Auth. Project
- Auth. Link Speed
- Auth. Ack
- ReAuthenticate
- Authenticated
- Prepare For Callback
- Wait For Modem Reset
- Wait For Callback
- Connected

**See also:**

RASDial

RASHangUp

## RASHangUp

**Description:** Terminates a RAS (remote access service) connection previously established using RASDial statement.

**Syntax:** RASHangUp connection

| Argument | Description |
|---|---|
| Connection | A number whose value is the connection handle returned by RASDial statement. |

Return value: None.

**See also:**

RASDial
RASGetStatus

# Remote Automation statements (RA for UNIX and Linux)

The following methods are supported for automating remote UNIX jobs

- **RAConnect**
- **RADisconnect**
- **RAGetWorkDir**
- **RADir**
- **RASetWorkDir**
- **RAFileDateTime**
- **RAFileSize**
- **RAFileTransfer**
- **RAFileReadAll**
- **RAFileSave**
- **RAFileOpen**
- **RAFileRead**
- **RAFileWrite**
- **RAFileClose**
- **RARun**
- **RARunAndWait**
- **RAProcessKill**

- **RAProcessList**

**Note:** For more information on these methods see manual for **24x7 Remote Automation Server for Linux and UNIX**

# Registry statements

## RegistryGetKey

**Description:** Gets a value from the Windows system registry.

**Syntax:** RegistryGetKey key, valuename, valuetype, return

| Argument | Description |
|---|---|
| key | A string whose value names the key in the system registry whose value you want to retrieve. To uniquely identify a key, specify the list of parent keys above it in the hierarchy, starting with the root key. The keys in the list are separated by backslashes |
| valuename | A string containing the name of a value in the registry. Each key can have one unnamed value and several named values. For the unnamed value, specify an empty string |
| valuetype | A string specifying the type of a value in the registry. The following types are supported:<br>• "STRING" — A null-terminated string<br>• "NUMBER" — A 32-bit number |
| return | A variable corresponding to the data type of valuetype that receives the returned value |

**Return value:** String or Number. The result type corresponds to the data type of valuetype.

**Usage:** A key is part of a tree of keys, descending from one of the predefined root keys. Each key is a subkey or child of the parent key above it in the hierarchy. There are four root strings:

- HKEY_CLASSES_ROOT
- HKEY_LOCAL_MACHINE
- HKEY_USERS
- HKEY_CURRENT_USER

A key is uniquely identified by the list of parent keys above it. The keys in the list are separated by slashes, as shown in these examples.
HKEY_LOCAL_MACHINE\Software\Microsoft\Access\7.0\Options
HKEY_CURRENT_USER\Control Panel\International

**See also:**

IniFileGetKey
IniFileSetKey
RegistrySetKey

# RegistrySetKey

**Description:** Sets the value for a key and value name in the system registry. If the key or value name does not exist, RegistrySetKey creates a new key or name and sets its value.

**Syntax:** RegistrySetKey key, valuename, valuetype, value

| Argument | Description |
|---|---|
| key | A string whose value names the key in the system registry whose value you want to set. To uniquely identify a key, specify the list of parent keys above it in the hierarchy, starting with the root key. The keys in the list are separated by backslashes. If key does not exist in the registry, RegistrySetKey creates a new key |
| valuename | A string containing the name of a value in the registry. Each key can have one unnamed value and several named values. For the unnamed value, specify an empty string |
| valuetype | A string specifying the type of a value in the registry. The following types are supported:<br>• "STRING" — A null-terminated string<br>• "NUMBER" — A 32-bit number |
| value | A constant or a variable corresponding to the data type of valuetype containing a value to be set in the registry |

**Return value:** None.

**Usage:** A key is part of a tree of keys, descending from one of the predefined root keys. Each key is a subkey or child of the parent key above it in the hierarchy. There are four root strings:

- HKEY_CLASSES_ROOT
- HKEY_LOCAL_MACHINE
- HKEY_USERS
- HKEY_CURRENT_USER

A key is uniquely identified by the list of parent keys above it. The keys in the list are separated by slashes, as shown in these examples.
HKEY_LOCAL_MACHINE\Software\Microsoft\Access\7.0\Options
HKEY_CURRENT_USER\Control Panel\International

**See also:**

IniFileGetKey
IniFileSetKey
RegistryGetKey

# RegistryDelete

**Description:** Deletes the specified registry key or value from the system registry. If the key contains values or other subkeys, RegistryDelete recursively deletes all of them.

**Syntax:** RegistryDelete key_or_value_name

| Argument | Description |
|---|---|
| key_or_value_name | A string whose value names the key or named value in the system registry that you want to delete. To uniquely identify a key, specify the list of parent keys above it in the hierarchy, starting with the root key. The keys in the list are separated by backslashes. |

**Return value:** None.

**Usage:** A key is part of a tree of keys, descending from one of the predefined root keys. Each key is a subkey or child of the parent key above it in the hierarchy. There are four root strings:

- HKEY_CLASSES_ROOT
- HKEY_LOCAL_MACHINE
- HKEY_USERS
- HKEY_CURRENT_USER

A key is uniquely identified by the list of parent keys above it. The keys in the list are separated by slashes, as shown in these examples.
HKEY_LOCAL_MACHINE\Software\Microsoft\Access\7.0\Options
HKEY_CURRENT_USER\Control Panel\International

**See also:**

RegistryList
RegistrySetKey
RegistryGetKey

# RegistryList

**Description:** Lists sub-keys or named values for the specified registry key

**Syntax:** RegistryList key, enumerator, return

| Argument | Description |
|---|---|
| key | A string whose value names the key in the system registry whose sub-keys or values you want to retrieve. To uniquely identify a key, specify the list of parent keys above it in the hierarchy, starting with the root key. The keys in the list are separated by backslashes. |
| enumerator | A string whose value must be one of the following: |

| | |
|---|---|
| | • SUBKEYS<br>• VALUES<br>Specify "SUBKEYS" value to enumerate sub-keys for the key, or specify VALUES to enumerate named values. |
| return | A string variable that receives the returned value. |

**Return value:** String. RegistryList returns comma-separated list of sub-keys or values.

**Usage:** A key is part of a tree of keys, descending from one of the predefined root keys. Each key is a subkey or child of the parent key above it in the hierarchy. There are four root strings:

- HKEY_CLASSES_ROOT
- HKEY_LOCAL_MACHINE
- HKEY_USERS
- HKEY_CURRENT_USER

A key is uniquely identified by the list of parent keys above it. The keys in the list are separated by slashes, as shown in these examples.
HKEY_LOCAL_MACHINE\Software\Microsoft\Access\7.0\Options
HKEY_CURRENT_USER\Control Panel\International

**See also:**

RegistryDelete
RegistrySetKey
RegistryGetKey

# Service statements

## ServiceContinue

**Description:** Resume the execution of a paused Windows NT (NT/2000/XP/2003/VISTA/2008/7) service

**Syntax:** ServicePause service

| Argument | Description |
|---|---|
| service | A string whose value is the name of the paused service that you want to resume. Service name is case insensitive. |

**Return value:** None.

**Usage:** This statement can be used on Windows NT (NT/2000/XP/2003/VISTA/2008/7) platform only. You must have sufficient authority to resume the specified service. You may want to check the service status before trying to stop it. An error occurs if the specified service has not been started. The ServiceGetStatus statement can be used to retrieve the status of a service.

**See also:**

ServicePause
ServiceStart
ServiceGetStatus

# ServiceGetStatus

**Description:** Retrieves the status of a Windows NT (NT/2000/XP/2003/VISTA/2008/7) service

**Syntax:** ServiceGetStatus service, return

| Argument | Description |
|---|---|
| service | A string whose value is the name of the service whose status you want to retrieve. Service name is case insensitive. |
| return | A string variable that receives the returned value |

**Return value:** String. The returned value can be one of the following:

| Value | Meaning |
|---|---|
| "STOP PENDING" | The service is stopping. |
| "STOPPED" | The service is not running. |
| "START PENDING" | The service is starting. |
| "RUNNING" | The service is running. |
| "CONTINUE PENDING" | The service continue is pending. |
| "PAUSE PENDING" | The service pause is pending. |
| "PAUSED" | The service is paused. |
| "ERROR" | An error occurred while querying service status. |

**Usage:** This statement can be used on Windows NT (NT/2000/XP/2003/VISTA/2008/7) platform only. You must have sufficient authority to query status of the specified service.

# ServicePause

**Description:** Pause the execution of a Windows NT (NT/2000/XP/2003/VISTA/2008/7) service

**Syntax:** ServicePause service

| Argument | Description |
|---|---|
| service | A string whose value is the name of the service that you want to suspend. Service name is case insensitive. |

**Return value:** None.

**Usage:** This statement can be used on Windows NT (NT/2000/XP/2003/VISTA/2008/7) platform only. You must have sufficient authority to pause the specified service. You may want to check the service status before trying to stop it. An error occurs if the specified service has not been started. The ServiceGetStatus statement can be used to retrieve the status of a service.

**See also:**

ServiceContinue
ServiceStart
ServiceGetStatus

# ServiceStart

**Description:** Starts the execution of a Windows NT (NT/2000/XP/2003/VISTA/2008/7) service

**Syntax:** ServiceStart service

| Argument | Description |
|---|---|
| service | A string whose value is the name of the service that you want to start. Service name is case insensitive. |

**Return value:** None.

**Usage:** This statement can be used on Windows NT (NT/2000/XP/2003/VISTA/2008/7) platform only. You must have sufficient authority to start the specified service. You may want to check the service status before trying to start it. An error occurs if the specified service is already running. The ServiceGetStatus statement can be used to retrieve the status of a service.

**See also:**

ServicePause
ServiceStop
ServiceGetStatus

# ServiceStop

**Description:** Stops the execution of a Windows NT (NT/2000/XP/2003/VISTA/2008/7) service

**Syntax:** ServiceStop service

| Argument | Description |
|---|---|
| service | A string whose value is the name of the service that you want to stop. Service name is case insensitive. |

**Return value:** None.

**Usage:** This statement can be used on Windows NT (NT/2000/XP/2003/VISTA/2008/7) platform only. You must have sufficient authority to stop the specified service. You may want to check the service status before trying to stop it. An error occurs if the specified service has not been started. The ServiceGetStatus statement can be used to retrieve the status of a service.

**See also:**

ServicePause
ServiceStart
ServiceGetStatus

# String statements

## Asc

**Description:** Converts the first character of a string to its ASCII number value.

**Syntax:** Asc ( string, result )

| Argument | Description |
|---|---|
| string | A string for which you want the ASCII value of the first character |
| result | A numeric variable that receives the returned value |

**Return value:** Number. Returns the ASCII value of the first character in string.

**Usage:** Use Asc to test the case of a character or manipulate text and letters. To find out the case of a character, you can check whether its ASCII value is within the appropriate range.

**See also:**

Char

# Char

**Description:** Converts a number to a string character.

**Syntax:** Char ( number, result )

| Argument | Description |
|---|---|
| number | A number you want to convert to a character |
| result | A string variable that receives the returned value |

**Return value:** String. Returns the string that consists of a single character whose ASCII value is number.

**See also:**

Asc

# Concat

**Description:** Appends one string to the end of another strings, in other words concatenates two strings.

**Syntax:** Concat s1, s2, return

| Argument | Description |
|---|---|
| s1 | The string to which you want to append the string s2 |
| s2 | The string you want to append to the end of the string s1 |
| return | A string variable that receives the returned value |

**Return value:** String. The concatenation operation returns the string that is union of s1 and s2.

**See also:**

ConcatEx

# ConcatEx

**Description:** Appends one or more strings to the end of each other, in other words concatenates them.

**Syntax:** ConcatEx s1, s2, s3, ...., return

| Argument | Description |
|---|---|
| s1 | The string to which you want to append the string s2 |
| s2 | The string you want to append to the end of the string s1 |
| s3 | The string you want to append to the end of the string s1 after s2 |
| return | A string variable that receives the returned value |

**Return value:** String. The concatenation operation returns the string that is union of s1 and s2 and so on.

**Usage:** Use ConcatEx instead of multiple Concat statement to concatenate multiple strings into a large one. Note that the JAL script engine automatically handles all datatype conversions so that if any of ConcatEx components is not a string it will be automatically converted to it. The number of components in one ConcatEx statement cannot exceed 32765.

**See also:**

Concat

## Fill

**Description:** Builds a string of the specified length by repeating the specified characters until the result string is long enough.

**Syntax:** Fill chars, n, return

| Argument | Description |
|---|---|
| chars | A string whose value will be repeated to fill the return string |
| n | A long whose value is the length of the string you want returned |
| return | A string variable that receives the returned value |

**Return value:** String. Returns a string n characters long filled with repetitions of the characters in the argument chars. If the argument chars has more than n characters, the first n characters of chars are used to fill the return string. If the argument chars has fewer than n characters, the characters in chars are repeated until the return string has n characters.

**See also:**

Space

# Format

**Description:** Formats data as a string according to a specified format mask. You can convert and format date, datetime, numeric, and time data. You can also apply a format to a string.

**Syntax:** Format ( data, format , result )

| Argument | Description |
|---|---|
| data | The data you want returned as a string with the specified formatting. Data can have a date, datetime, numeric, time, or string data type |
| format | A string of the masks you want to use to format the data. The masks consist of formatting information specific to the data type of data. The format string can consist of more than one mask, depending on the data type of data. Each mask is separated by a semicolon. |
| result | A string variable that receives the returned value |

**Return value:** String. Returns data in the specified format if it succeeds and the empty string ("") if the data type of data does not match the type of mask specified or format is not a valid mask.

**Usage:**
Formats for date, datetime, string, and time data can include one mask only. Formats for numeric data can have up to three masks. A format with a single mask handles both positive and negative data. If there are additional masks, the first mask is for positive values, and the additional masks are for negative, and zero values. If the format doesn't match the data type, the 24x7 Scheduler will try to apply the mask, which can produce unpredictable results.

**See also:**

Format symbols

String statement

# GetToken

**Description:** Obtains first token from the specified string.

**Syntax:** GetToken s1, s2, return

| Argument | Description |
|---|---|
| s1 | A string that is token separator in the string s1. This could be a single-character string such as space or a multi-character string. |
| s2 | A string from which you want to extract the first available token |
| return | A string variable that receives the returned value |

**Return value:** String. The first token from string s2 separated by s1. If no separator is found in the string s2 then GetToken returns the entire string s2.

**Usage:** When the first argument s2 is a variable then GetToken also removes the first found token and the following separator from the string s2 so that it begins with the next token. When there is no more tokens available then GetToken returns the empty string (""). You can call GetToken in a loop to extract all tokens from the string s2.

## InStr

**Description:** Finds one string within another string.

**Syntax:** InStr string1, string2, start, return

| Argument | Description |
|---|---|
| string1 | The string in which you want to find string2 |
| string2 | The string you want to find in string1 |
| start | The number indicating starting position where the search will begin in string1. |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns a number whose value is the starting position of the first occurrence of string2 in string1 after the position specified in start. If string2 is not found in string1 or if start is not within string1, InStr returns 0.

**Usage:** The InStr function is case sensitive. Alternatively, you can use Pos statements which syntax and function is the same.

**See also:**

Pos

## Left

**Description:** Obtains a specified number of characters from the beginning of a string.

**Syntax:** Left string, n, return

| Argument | Description |
|---|---|
| string | The string containing the characters you want |
| n | The number of characters you want |
| return | A string variable that receives the returned value |

**Return value:** String. Returns the leftmost n characters in string if it succeeds and the empty string ("") if an error occurs. If n is greater than or equal to the length of the string, Left returns the entire string. It does not add spaces to make the return value's length equal to n.

**See also:**

Right

Mid

## Length

**Description:** Reports the length of a string.

**Syntax:** Len string, return

| Argument | Description |
|---|---|
| String | The string for which you want the length |
| Return | A numeric variable that receives the returned value |

**Return value:** Number. Returns a long containing the length of string if it succeeds and -1 if an error occurs.

## Lower

**Description:** Converts all the characters in a string to lowercase.

**Syntax:** Lower string, return

| Argument | Description |
|---|---|
| string | The string you want to convert to lowercase letters |
| return | A string variable that receives the returned value |

**Return value:** String. Returns string with uppercase letters changed to lowercase if it succeeds and the empty string ("") if an error occurs.

**See also:**

Upper

# LTrim

**Description:** Removes leading spaces from the beginning of a string.

**Syntax:** LTrim string, return

| Argument | Description |
|---|---|
| string | The string you want returned with leading blanks deleted |
| return | A string variable that receives the returned value |

**Return value:** String. Returns a copy of string with leading blanks deleted if it succeeds and the empty string ("") if an error occurs.

**See also:**

RTrim
Trim

# Match

**Description:** Determines whether a string's value contains a particular pattern of characters.

**Syntax:** Match string, textpattern, return

| Argument | Description |
|---|---|
| string | The string in which you want to look for a pattern of characters |
| textpattern | A string whose value is the text pattern |
| return | A boolean variable that receives the returned value |

**Return value:** Boolean. Returns TRUE if string matches textpattern and FALSE if it does not. Match also returns FALSE if either argument has not been assigned a value or the pattern is invalid.

**Usage:** Match enables you to evaluate whether a string contains a general pattern of characters. To find out whether a string contains a specific substring, use the Pos statement.
Textpattern is similar to a regular expression. It consists of **metacharacters**, which have special meaning, and ordinary characters, which match themselves. You can specify that the string begin or end with one or more characters from a set, or that it contains any characters except those in a set.

A text pattern consists of **metacharacters**, which have special meaning in the match string, and **non-metacharacters**, which match the characters themselves.

**See also:**

metacharacters and sample patterns

# Mid

**Description:** Obtains a specified number of characters from a specified position in a string.

**Syntax:** Mid string, start, length, return

| Argument | Description |
|---|---|
| string | The string from which you want characters returned |
| start | A number specifying the position of the first character you want returned. (The position of the first character of the string is 1) |
| length | A number whose value is the number of characters you want returned. If length is greater than the number of characters to the right of start, Mid returns the remaining characters in the string |
| return | A string variable that receives the returned value |

**Return value:** String. Returns characters specified in length of string starting at character start. If start is greater than the number of characters in string, the Mid function returns the empty string (""). If length is greater than the number of characters remaining after the start character, Mid returns the remaining characters. The return string is not filled with spaces to make it the specified length.

**See also:**

Left
Right

# Number

**Description:** Converts a string to a number.

**Syntax:** Number ( string, result )

| Argument | Description |
|---|---|
| string | A string you want returned as a number |
| result | A numeric variable that receives the returned value |

**Return value:** Number. Returns the contents of string as a number. If string is not a valid number, Number returns 0.

**See also:**

String
Format

# Pos

**Description:** Finds one string within another string.

**Syntax:** Pos string1, string2, start, return

| Argument | Description |
|---|---|
| string1 | The string in which you want to find string2 |
| string2 | The string you want to find in string1 |
| start | The number indicating starting position where the search will begin in string1. |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Returns a number whose value is the starting position of the first occurrence of string2 in string1 after the position specified in start. If string2 is not found in string1 or if start is not within string1, Pos returns 0.

**Usage:** The Pos function is case sensitive. Alternatively, you can use InStr statements which syntax and function is the same.

**See also:**

InStr

# Replace

**Description:** Replaces a portion of one string with another.

**Syntax:** Replace string1, start, n, string2, return

| Argument | Description |
|---|---|
| string1 | The string in which you want to replace characters with string2 |
| start | A long whose value is the number of the first character you want replaced. (The first character in the string is number 1) |
| N | A number whose value is the number of characters you want to replace |
| string2 | The string that will replace characters in string1. The number of characters in string2 can be greater than, equal to, or fewer than the number of characters you are |

| | |
|---|---|
| | replacing |
| return | A string variable that receives the returned value |

**Return value:** String. Returns the string with the characters replaced if it succeeds and the empty string ("") if it fails.

**Usage:**
If the start position is beyond the end of the string1, Replace appends string2 to string1. If there are fewer characters after the start position than specified in n, Replace replaces all the characters to the right of character start. If n is zero, then in effect Replace inserts string2 into string1.

**See also:**

Concat

# Reverse

**Description:** Reverses the order or characters in a string.

**Syntax:** Reverse string, return

| Argument | Description |
|---|---|
| string | A string whose characters you want to reorder so that the last character is first and the first character is last |
| return | A string variable that receives the returned value |

**Return value:** String. Returns a string with the characters of string in reversed order if it succeeds and the empty string ("") if an error occurs.

# Right

**Description:** Obtains a specified number of characters from the end of a string.

**Syntax:** Right string, n, return

| Argument | Description |
|---|---|
| string | The string containing the characters you want |
| n | A number specifying the number of characters you want |
| return | A string variable that receives the returned value |

**Return value:** String. Returns the rightmost n characters in string if it succeeds and the empty string ("") if an error occurs. If n is greater than or equal to the length of the string, Right returns the entire string. It does not add spaces to make the return value's length equal to n.

**See also:**

Left
Mid

# RTrim

**Description:** Removes spaces from the end of a string.

**Syntax:** RTrim string, return

| Argument | Description |
|---|---|
| string | The string you want returned with trailing blanks deleted |
| return | A string variable that receives the returned value |

**Return value:** String. Returns a copy of string with trailing blanks deleted if it succeeds and the empty string ("") if an error occurs.

**See also:**

LTrim
Trim

# Space

**Description:** Builds a string of the specified length whose value consists of spaces.

**Syntax:** Space n, return

| Argument | Description |
|---|---|
| N | A number whose value is the length of the string you want filled with spaces |
| Return | A string variable that receives the returned value |

**Return value:** String. Returns a string filled with n spaces if it succeeds and the empty string ("") if an error occurs.

**See also:**

Fill

# String

**Description:** Converts data to a string. You can convert and format date, datetime, numeric, and time data.

**Syntax:** String ( data, result )

| Argument | Description |
|---|---|
| data | The data you want returned as a string with the specified formatting. Data can have a date, datetime, numeric, time, or string data type |
| result | A string variable that receives the returned value |

**Return value:** String. Returns the string based on the data value.

**Usage:**
For date, datetime, numeric, and time data, the 24x7 Scheduler uses the system's default format for the returned string. You can use the Format statement to converting data to a string using user-defined format mask.

**See also:**

Format

# Trim

**Description:** Removes leading and trailing spaces from a string.

**Syntax:** Trim string, return

| Argument | Description |
|---|---|
| string | The string you want returned with leading and trailing spaces deleted |
| return | A string variable that receives the returned value |

**Return value:** String. Returns a copy of string with all leading and trailing spaces deleted if it succeeds and the empty string ("") if an error occurs.

**See also:**

RTrim
Ltrim

## Upper

**Description:** Converts all the characters in a string to uppercase.

**Syntax:** Upper string, return

| Argument | Description |
|---|---|
| string | The string you want to convert to uppercase letters |
| return | A string variable that receives the returned value |

**Return value:** String. Returns string with lowercase letters changed to uppercase if it succeeds and the empty string ("") if an error occurs.

**See also:**

Lower

# Telnet and Secure Shell statements

The same statements can be used for both Telnet and Secure Shell automation. Call the TelnetConfig statement in the beginning of a job to specify which protocol to use.

## TelnetClose

**Description:** Closes active Telnet, Rlogin or Secure Shell session.

**Syntax:** TelnetClose

**Usage**: This statement has no arguments. TelnetClose closes session that you previously opened using TelnetOpen statement.

**See also:**

TelnetOpen

# TelnetOpen

**Description:** Opens new Telnet, Rlogin or Secure Shell session.

**Syntax:** TelnetOpen host, user, password

| Argument | Description |
|---|---|
| Host | A string whose value is the host name of the remote computer (for example, mycompany.com) or the IP number of the site in ASCII dotted-decimal format (for example, 11.0.1.45) |
| User | A string whose value is the name of the user to log on |
| Password | A string whose value is the password to use to log on |

**Return value:** none.

**Usage:** One job may have only one Telnet session open at a time. However, multiple jobs may have multiple Telnet sessions opened simultaneously.

To execute commands using Secure Shell protocol call TelnetConfig "TELNET PROTOCOL", "SECURE" before calling TelnetOpen statement.

TelnetOpen statement opens new connection to the remote host then attempts to intercept the prompts for user name and password. By default it expects the host to prompt for the user name in the form of "**login:**". After the prompt for user is received and the name is provided, it expects the host to prompt for the password in the form of "**password:**"

If the host uses different prompts, call TelnetConfig statement to change the default prompts before you call TelnetOpen statement.

Host can be any computer using any Operation System that supports standard Telnet protocol. Telnet protocol is supported by most popular Operation Systems including UNIX, Windows NT, Windows 2000, Windows XP, mainframe computers, etc. **In order to connect to the host and execute any valid host commands you do not need 24x7 Scheduler or 24x7 Remote Agent running on the host computer.**

 **Tip:** To troubleshoot connection problems use TelnetConfig statement to unhide the hidden Telnet terminal window, which displays all communication messages and errors. After you are done with setup and debugging of your script, comment out or remove the line with TelnetConfig statement un-hiding the terminal.

**See also:**

- TelnetConfig
- TelnetClose
- TelnetSend
- TelnetReceive

# TelnetSend

**Description:** Sends a command or a data to the remote host using active Telnet, Rlogin or Secure Shell session.

**Syntax:** TelnetSend text

| Argument | Description |
|---|---|
| Text | A string whose value you want to send as a command or data to the remote host |

**Return value:** none.

**Usage:** TelnetSend statement simulates a user manually typing some text in the standard Telnet terminal window then pressing the Enter key to submit the entered command or data. TelnetSend statement automatically adds carriage return (CR) character (ASCII code 13). If the text is sent as a command and it contains one or more CR characters then every part of the text separated by the CR will be considered by host as a separate command.

 **Tips:**

- To troubleshoot various Telnet problems use TelnetConfig statement to unhide the hidden Telnet terminal window, which displays all communication messages and errors. After you are done with setup and debugging of your script, comment out or remove the line with TelnetConfig statement un-hiding the terminal.
- To execute commands using Secure Shell protocol call TelnetConfig "TELNET PROTOCOL", "SECURE" before calling TelnetOpen statement.

**See also:**

TelnetConfig
TelnetOpen
TelnetReceive

# TelnetReceive

**Description:** Returns buffered data received from the remote host using active Telnet, Rlogin or Secure Shell session.

**Syntax:** TelnetReceive return

| Argument | Description |
|---|---|
| return | A string variable that receives the returned value. |

**Return value:** String. Returns all data received from the host computer since last TelnetSend ot TelnetReceive operation.

**Usage:** 24x7 Scheduler stores all text data it receives from the host computer in an internal buffer. Every call to TelnetReceive statement simply returns data from the buffer. It is up to you how you interpret or use this data. If there is no data available in the buffer, TelnetReceive will pause the script execution until new data is received from the server or timeout occurs. You can use TelnetConfig statement to change the value of the default timeout period.

 **Important Note:** Every call to TelnetSend or TelnetReceive statements causes the internal buffer to be reset (cleared).

**See also:**

TelnetConfig
TelnetOpen
TelnetSend

# TelnetConfig

**Description:** Changes various settings for the active or pending Telnet, Rlogin and Secure Shell sessions.

**Syntax:** TelnetConfig property, new_value

| Argument | Description |
|---|---|
| Property | A string whose value is the name of the property for the Telnet session that you want to change. The following properties are supported:<br>• "PROTOCOL" – controls which communication protocol to use. Supported protocols are "TELNET" or "RLOGIN". To use Secure Telnet communication mode (also called **Secure Shell** or simply SSH protocol) use "TELNET PROTOCOL" property. The default value is *"TELNET."*<br>• "LOGIN PROMPT" – controls the login prompt that 24x7 Scheduler should recognize during the authentication process while connecting with your Telnet server. The default value is *"login:"*<br>This option is used only with the telnet protocol.<br>• "PASSWORD PROMPT" – controls the password prompt that 24x7 Scheduler should recognize during the authentication process while connecting with your Telnet server. The default value is *"password:"*<br>This option is used only with the telnet protocol.<br>• "TIMEOUT" – controls communication timeout settings. Specify new_value in seconds. The default value is 10 seconds.<br>• "TERMINAL" – controls whether to show or hide the Telnet terminal, which can be used for debugging of Telnet operations. Specify either "SHOW" or "HIDE" for the new_value |

| | |
|---|---|
| | • "TELNET PROTOCOL" – controls which kind of Telnet protocol to use. Specify either "SECURE" or "NON SECURE". The default value is "NON SECURE".<br>• "PORT" – controls which port to use for communications. Default values are different for different protocols:<br>telnet – 23<br>rlogin – 513<br>secure shell (SSH) – 22<br>• "AUTHENTICATE" – controls whether 24x7 Scheduler should perform user authentication on the connected server. Specify either "TRUE" or "FALSE". The default value is "TRUE". This option is used only with the telnet protocol.<br>• "EOL" – controls which symbol or symbols to use for end-of-line when sending commands to server. The default value is carriage return (CR) character (ASCII code 13). |
| New_value | A string whose value is the new value for the property that you want to change. |

**Return value:** None.

**Usage:** Use TelnetConfig statement to configure and troubleshoot Telnet sessions. Both parameter names and their values are case insensitive.

Do not mistake secure telnet protocol (also called secure shell or SSH) and telnet type communications over SSL channel.

24x7 Scheduler supports both old and new versions of the secure shell protocols: SSH1 and SSH2

If you want to use communication protocol other than "TELNET" always call TelnetConfig with the "PROTOCOL" parameter before changing setting for communication port and secure mode.

**Tips:**
- To unhide the full featured Telnet terminal window use TelnetConfig "TERMINAL", "SHOW".
- To change the default logon prompt to "user name:" use TelnetConfig "LOGIN PROMPT", "user name:" before you call TelnetOpen statement.
- If you are using slow connection to the remote host, increase the timeout value, for example TelnetConfig "TIMEOUT", "30".
- To execute commands using secure shell protocol first call TelnetConfig "TELNET PROTOCOL", "SECURE" and then call other Telnet commands TelnetOpen, TelnetSend, TelnetRead and TelnetClose as needed for the processing. These commands will be executed using secure shell SSH protocol.

**See also:**
TelnetOpen
TelnetSend

# Window statements

## WindowActivate

**Description:** Puts the process that created the specified window into the foreground and activates the window. Keyboard input is directed to the window, and various visual cues are changed for the user.

**Syntax:** WindowActivate handle

| Argument | Description |
|---|---|
| handle | A number whose value is the handle of the window that you want to send in front of other windows and make activate. |

**Return value:** None.

**Usage:** Use WindowActivate statement before sending keystrokes to desired window.

**See also:**

WindowFind
WindowGetActive
SendKeys

## WindowClickButton

**Description:** Searches the specified window for a control having the specified caption and simulates mouse click on the found control.

**Syntax:** WindowClickButton handle, caption

| Argument | Description |
|---|---|
| handle | A number whose value is the handle of the window that owns the control with the specified caption. Controls are buttons, pictures, radio-buttons, check-boxes, etc… |
| caption | A string whose value is the caption of the control that you want to “click" . |

**Return value:** None.

**Usage:** If the specified window is a parent of other windows, WindowClickButton automatically searches all child windows. It “clicks" the first control that has the specified caption. Use WindowFind or any other appropriate Window statement to retrieve handle of the desired parent window.

**See also:**

WindowGetActive
WindowFind
SendKeys

# WindowClose

**Description:** Closes the specified window.

**Syntax:** WindowClose handle

| Argument | Description |
|---|---|
| handle | A number whose value is the handle of the window that you want to close |

**Return value:** None.

**Usage:** If the specified window is a parent of other windows, WindowClose automatically closes children before parent. However, there is no guarantee that the specified window will be closed. In some situations when there are some changes pending, the application that owns the specified window may invoke a dialog box, which prompts the user to save changes before exiting. Any processing that the application does on exit will be carried out as usual.

**Notes:**

- Often closing of the main window causes that process to finish. It is highly recommended to use such method instead of using ProcessKill statement.
- Sending the ALT+F4 keystroke to the active window in most cases causes window closing as well as WindowClose. The CTRL+F4 keystroke closes sheets in multi-document interface systems such as MS Excel.

**See also:**

WindowWaitClose
WindowFind
SendKeys

# WindowFind

**Description:** Retrieves the handle of the window whose title matches the specified strings. The specified strings may include the percent (%) sign to indicate a wildcard.

**Syntax:** WindowFind title, return

| Argument | Description |
|---|---|
| title | A string whose value is the full or partial title of the window that you want to find |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Return a handle of the first found window whose title matches the specified string.

**Usage:** Use search with a percent (%) sign appended to the end of the search string when the window title varies. The percent sign is used as a wildcard that matches any group of characters. It solves problems with applications that change their titles depending on the loaded document. For example a search for "Microsoft Word %" will find an instance of Microsoft Word with any document loaded.

**Note:**
WindowFind searches all windows including invisible windows. Search can be performed only for windows that run on the same Desktop.

**See also:**

- WindowGetActive
- WindowGetChild
- WindowGetParent
- WindowGetProcess
- WindowGetTitle
- SendKeys

## WindowGetActive

**Description:** Obtains the handle of the active window. An active window is the window that receives keyboard input.

**Syntax:** WindowGetActive return

| Argument | Description |
|---|---|
| return | A numeric variable that receives the returned value |

**Return value:** Number. Return a handle of the active window.

**See also:**

- WindowFind
- WindowGetProcess
- WindowGetTitle

# WindowGetChild

**Description:** Obtains the handle of the first child window for the specified parent window.

**Syntax:** WindowGetChild handle, return

| Argument | Description |
|---|---|
| handle | A number whose value is the handle of a parent window which owns the window that you want to obtain |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Return a handle of the child window.

**Usage:** WindowGetChild returns a handle of the first child window reported by Windows. You can use WindowGetNext and WindowGetPrevious statements to obtains handles of other child windows for the same parent window. If the statement succeeds, the return value is a window handle. If no child window exists for the specified window, the return value is 0.

**See also:**

WindowFind
WindowGetTitle
WindowGetFirst
WindowGetLast
WindowGetNext
WindowGetParent
WindowGetPrevious

# WindowGetFirst

**Description:** Obtains the handle of the first window that has the same type as the specified window.

**Syntax:** WindowGetFirst handle, return

| Argument | Description |
|---|---|
| handle | A number whose value is the handle of the window for which you want to obtain the first window of the same type |
| return | A numeric variable that receives the returned value |

**Return value:** Number. The retrieved handle identifies the window of the same type that is highest in the Z order. If the specified window is a topmost window, the handle identifies the topmost window that is highest in the Z order. If the specified window is a top-level window, the handle identifies the top-level window that is highest in the Z order. If the specified window is a child window, the handle identifies the sibling window that is highest in the Z order. If the statement succeeds, the return value is a window handle. If no window exists with the specified relationship to the specified window, the return value is 0.

**See also:**

- WindowFind
- WindowGetTitle
- WindowGetChild
- WindowGetLast
- WindowGetNext
- WindowGetParent
- WindowGetPrevious

# WindowGetLast

**Description:** Obtains the handle of the last window that has the same type as the specified window.

**Syntax:** WindowGetLast handle, return

| Argument | Description |
|---|---|
| handle | A number whose value is the handle of the window for which you want to obtain the last window of the same type |
| return | A numeric variable that receives the returned value |

**Return value:** Number. The retrieved handle identifies the window of the same type that is lowest in the Z order. If the specified window is a topmost window, the handle identifies the topmost window that is lowest in the Z order. If the specified window is a top-level window, the handle identifies the top-level window that is lowest in the Z order. If the specified window is a child window, the handle identifies the sibling window that is lowest in the Z order. If the statement succeeds, the return value is a window handle. If no window exists with the specified relationship to the specified window, the return value is 0.

**See also:**

- WindowFind
- WindowGetTitle
- WindowGetFirst
- WindowGetChild
- WindowGetNext
- WindowGetParent
- WindowGetPrevious

# WindowGetNext

**Description:** Obtains the handle of the next window that has the same type as the specified window.

**Syntax:** WindowGetNext handle, return

| Argument | Description |
|---|---|
| handle | A number whose value is the handle of the window for which you want to obtain the next window of the same type |
| return | A numeric variable that receives the returned value |

**Return value:** Number. The retrieved handle identifies the window of the same type that is next in the Z order. If the specified window is a topmost window, the handle identifies the topmost window that is next in the Z order. If the specified window is a top-level window, the handle identifies the top-level window that is next in the Z order. If the specified window is a child window, the handle identifies the sibling window that is next in the Z order. If the statement succeeds, the return value is a window handle. If no window exists with the specified relationship to the specified window, the return value is 0.

**See also:**

WindowFind
WindowGetTitle
WindowGetFirst
WindowGetLast
WindowGetChild
WindowGetParent
WindowGetPrevious

# WindowGetParent

**Description:** Obtains the handle of the parent window for the specified child window.

**Syntax:** WindowGetParent handle, return

| Argument | Description |
|---|---|
| handle | A number whose value is the handle of a child window for which you want to obtain it's parent window |
| return | A numeric variable that receives the returned value |

**Return value:** Number. The retrieved handle identifies the specified window's owner window. If no parent window for the specified window, the return value is 0.

**See also:**

WindowFind
WindowGetTitle
WindowGetFirst
WindowGetLast
WindowGetNext
WindowGetPrevious
WindowGetChild

# WindowGetPrevious

**Description:** Obtains the handle of the previous window that has the same type as the specified window.

**Syntax:** WindowGetPrevious handle, return

| Argument | Description |
|---|---|
| handle | A number whose value is the handle of the window for which you want to obtain the previous window of the same type |
| return | A numeric variable that receives the returned value |

**Return value:** Number. The retrieved handle identifies the window of the same type that is previous in the Z order. If the specified window is a topmost window, the handle identifies the topmost window that is previous in the Z order. If the specified window is a top-level window, the handle identifies the top-level window that is previous in the Z order. If the specified window is a child window, the handle identifies the sibling window that is previous in the Z order. If the statement succeeds, the return value is a window handle. If no window exists with the specified relationship to the specified window, the return value is 0.

**See also:**

- WindowFind
- WindowGetTitle
- WindowGetFirst
- WindowGetLast
- WindowGetNext
- WindowGetParent
- WindowGetChild

# WindowGetProcess

**Description:** Obtains process ID for the specified window.

**Syntax:** WindowGetProcess handle, return

| Argument | Description |
|---|---|
| handle | A number whose value is the handle of the window for which you want to find its process ID |
| return | A numeric variable that receives the returned value |

**Return value:** Number. Return process ID for the specified window.

**Usage:** Use WindowGetProcess to pass returned value to the ProcessKill statement when you cannot gracefully close that process and want to terminate the process anyway.

**See also:**

ProcessGetWindow
ProcessKill
ProcessGetID
WindowClose
WindowFind
WindowGetActive

# WindowGetTitle

**Description:** Obtains the title of the specified window.

**Syntax:** WindowGetTitle handle, return

| Argument | Description |
|---|---|
| handle | A number whose value is the handle of the window whose title you want to obtain |
| return | A string variable that receives the returned value |

**Return value:** String. Returns window title for the specified window. If the specified window handle is invalid, the return value is empty string (“”).

**See also:**

WindowFind
WindowGetActive

# WindowWaitClose

**Description:** Waits for the specified window to disappear (close). The specified window title may include the percent (%) sign to indicate a wildcard.

**Syntax:** WindowWaitClose title, timeout

| Argument | Description |
|---|---|
| title | A string whose value is the full or partial title of the window that you want to find |
| timeout | A number whose value is the maximum time interval within which you allow the specified window to close. Use 0 timeout to allow infinite waiting. |

**Return value:** None.

**Usage:** Execution of the script will not continue until the window with the specified title text is no longer present or timeout occurs. The window title may contain the percent (%) symbol at the end to indicate a wildcard. Use it when the desired window title varies. This solves the problem with applications that change their titles depending on the document loaded. The percent sign matches any group of characters. For example a search for "Microsoft Word %"will find an instance of Microsoft Word with any document loaded.

**Note:**
WindowWaitClose checks both visible and invisible windows

**See also:**

WindowWaitOpen
WindowFind
WindowClose

# WindowWaitOpen

**Description:** Waits for the specified window to appear (open). The specified window title may include the percent (%) sign to indicate a wildcard.

**Syntax:** WindowWaitOpen title, timeout

| Argument | Description |
|---|---|
| title | A string whose value is the full or partial title of the window that you want to find |
| timeout | A number whose value is the maximum time interval within which you allow the specified window to open. Use 0 timeout to allow infinite waiting. |

**Return value:** None.

**Usage:** Execution of the script will not continue until the window with the specified title text is found or timeout occurs. The window title may contain the percent (%) symbol at the end to indicate a wildcard. Use it when the desired window title varies. This solves the problem with applications that change their titles depending on the document loaded. The percent sign matches any group of characters. For example a search for "Microsoft Word %"will find an instance of Microsoft Word with any document loaded.

**Note:**
WindowWaitOpen checks both visible and invisible windows.

**See also:**

WindowWaitClose
WindowFind

# WindowPostMessage

**Description:** Adds the specified Windows message to the message queue for the specified window.

**Syntax:** WindowPostMessage handle, message, word, long

| Argument | Description |
|---|---|
| handle | A number whose value is the system handle of a window to which you want to post a message |
| message | A number whose value is the system message number of the message you want to send |
| word | A number whose value is the word value of the message. If this argument is not used by the message, use 0 |
| long | A number whose value is the long value of the message. If this argument is not used by the message, use 0 |

**Return value:** None.

**Usage:** WindowPostMessage statement posts the message identified by message and optionally, word and long, to the window identified by handle using Windows SDK function PostMessage. The message is added to the end of the object's message queue. Opposite to WindowSendMessage statement, WindowPostMessage is asynchronous; it does not stop execution of the script until the message is processed.

 **Note:**
See Windows Software Development Kit (SDK) documentation for complete information on supported messages and parameters

**See also:**

WindowSendMessage

Call

# WindowSendMessage

**Description:** Sends the specified Windows message to the specified window so that it is executed immediately.

**Syntax:** WindowSendMessage handle, message, word, long

| Argument | Description |
|---|---|
| handle | A number whose value is the system handle of a window to which you want to send a message |
| message | A number whose value is the system message number of the message you want to send |
| word | A number whose value is the word value of the message. If this argument is not used by the message, use 0 |

| | |
|---|---|
| long | A number whose value is the long value of the message. If this argument is not used by the message, use 0 |

**Return value:** None.

**Usage:** WindowSendMessage statement sends the message identified by message and optionally, word and long, to the window identified by handle using Windows SDK function SendMessage. The message is sent directly to the object, bypassing the object's message queue. Execution of the script will not continue until the message is processed.

 **Note:**
See Windows Software Development Kit (SDK) documentation for complete information on supported messages and parameters

**See also:**

WindowPostMessage

Call

# Functions in database filter and sort expressions

You can use the following functions in sort and filter expressions. The functions are organized in the following categories.

- Data type checking and conversion functions
- Day, date, and time functions
- Miscellaneous functions
- Numeric functions
- String functions

**See also:**

Operators

JAL Control-of-Flow Statements

JAL Statements by Category

## Data type checking fnctions

Asc
Char
Date
DateTime
Integer
IsDate
IsNull
IsNumber
IsTime
Long
Number
String
Time

**See also:**

Functions by category

## IsDate

**Description:** Tests whether a string value is a valid date.

**Syntax:** IsDate ( string )

| Argument | Description |
|---|---|
| string | A string whose value you want to test to determine whether it is a valid date |

**Return value:** Boolean. Returns TRUE if datevalue is a valid date and FALSE if it is not.

## IsNull

**Description:** Reports whether the value of a column or expression is NULL.

**Syntax:** IsNull ( any )

| Argument | Description |
|---|---|
| any | A column or expression that you want to test to determine whether its value is NULL |

**Return value:** Boolean. Returns TRUE if any is NULL and FALSE if it is not.

**Usage:** Use IsNull to test whether a user-entered value or a value retrieved from the database is NULL.

## IsNumber

**Description:** Reports whether the value of a string is a number.

**Syntax:** IsNumber ( string )

| Argument | Description |
|---|---|
| string | A string whose value you want to test to determine whether it is a valid number |

**Return value:** Boolean. Returns TRUE if string is a valid number and FALSE if it is not.

## IsTime

**Description:** Reports whether the value of a string is a valid time value.

**Syntax:** IsTime (timevalue)

| Argument | Description |
|---|---|
| timevalue | A string whose value you want to test to determine whether it is a valid time |

**Return value:** Boolean. Returns TRUE if timevalue is a valid time and FALSE if it is not.

# Date and time functions

## Date

**Description:** Converts a string whose value is a valid date to a value of data type date.

**Syntax:** Date ( string )

| Argument | Description |
|---|---|
| string | A string containing a valid date (such as Jan 1, 1998, or 12-31-99) that you want returned as a date |

**Return value:** Date. Returns the date in string as a date. If string does not contain a valid date, Date returns NULL.

**Usage:** The value of the string must be a valid date.
Valid dates can include any combination of day (1-31), month (1-12 or the name or abbreviation of a month), and year (two or four digits). Leading zeros are optional for month and day. If the month is a name or an abbreviation, it can come before or after the day; if it is a number, it must be in the month location specified in the Windows control panel. A four-digit number is assumed to be a year.
If the year is two digits, then program chooses the century, as follows. If the year is between 00 and 49, program assumes 20 as the first two digits; if it is between 50 and 99, 24x7 Scheduler assumes 19. If your data includes dates before 1950, such as birth dates, always specify a four-digit year so that 24x7 Scheduler interprets the date as intended.

24x7 Scheduler handles years from 1000 to 3000 inclusive.

An expression has a more limited set of data types than the functions that can be part of the expression. Although the Date function returns a date value, the whole expression is promoted to a DateTime value. Therefore, if your expression consists of a single Date function, it will appear that Date returns the wrong data type. To convert the date without the time, choose an appropriate convert format.

# DateTime

**Description:** Combines a date and a time value into a DateTime value.

**Syntax:** DateTime ( date {, time } )

| Argument | Description |
|---|---|
| date | A valid date (such as Jan 1, 1998, or 12-31-99) or a blob variable whose first value is a date that you want included in the value returned by DateTime |
| Time (optional) | (optional) A valid time (such as 8AM or 10:25:23:456799) or a blob variable whose first value is a time that you want included in the value returned by DateTime. If you include a time, only the hour portion is required. If you omit the minutes, seconds, or microseconds, they are assumed to be 0s. If you omit AM or PM, the hour is determined according to the 24-hour clock |

**Return value:** DateTime. Returns a DateTime value based on the values in date and optionally time. If time is omitted, DateTime uses 00:00:00.000000 (midnight).

**Usage:** For information on valid dates, see Date function.

# Day

**Description:** Obtains the day of the month in a date value.

**Syntax:** Day (date )

| Argument | Description |
|---|---|
| date | The date from which you want the day |

**Return value:** Integer. Returns an integer (1-31) representing the day of the month in date.

# DayName

**Description:** Determines the day of the week in a date value and returns the weekday's name.

**Syntax:** DayName (date )

| Argument | Description |
|---|---|
| date | The date for which you want the name of the day |

**Return value:** String. Returns a string whose value is the name of the weekday (Sunday, Monday, and so on) for date.

# DayNumber

**Description:** Determines the day of the week of a date value and returns the number of the weekday.

**Syntax:** DayNumber (date )

| Argument | Description |
|---|---|
| date | The date from which you want the number of the day of the week |

**Return value:** Integer. Returns an integer (1-7) representing the day of the week of date. Sunday is day 1, Monday is day 2, and so on.

# DaysAfter

**Description:** Determines the day of the week of a date value and returns the number of the weekday.

**Syntax:** DaysAfter ( date1, date2 )

| Argument | Description |
|---|---|
| date1 | A date value that is the start date of the interval being measured |
| date2e | A date value that is the end date of the interval |

**Return value:** Long. Returns a long containing the number of days date2 occurs after date1. If date2 occurs before date1, DaysAfter returns a negative number.

## Hour

**Description:** Obtains the hour in a time value. The hour is based on a 24-hour clock.

**Syntax:** Hour ( time )

| Argument | Description |
|---|---|
| Time | The time value from which you want the hour |

**Return value:** Integer. Returns an integer (00-23) containing the hour portion of time.

## Minute

**Description:** Obtains the number of minutes in the minutes portion of a time value.

**Syntax:** Minute ( time )

| Argument | Description |
|---|---|
| Time | The time value from which you want the minutes |

**Return value:** Integer. Returns the minutes portion of time (00 to 59).

## Month

**Description:** Determines the month of a date value.

**Syntax:** Month ( date )

| Argument | Description |
|---|---|
| date | The date from which you want the month |

**Return value:** Integer. Returns an integer (1 to 12) whose value is the month portion of date.

## Now

**Description:** Obtains the current time based on the system time of the client machine.

**Syntax:** Now ( )

**Return value:** Time. Returns the current time based on the system time of the client machine.

## RelativeDate

**Description:** Obtains the date that occurs a specified number of days after or before another date.

**Syntax:** RelativeDate ( date, n )

| Argument | Description |
|---|---|
| date | A date value |
| n | An integer indicating the number of days |

**Return value:** Date. Returns the date that occurs n days after date if n is greater than 0. Returns the date that occurs n days before date if n is less than 0.

## RelativeTime

**Description:** Obtains the time that occurs a specified number of seconds after or before another time.

**Syntax:** RelativeTime ( time, n )

| Argument | Description |
|---|---|
| Time | A time value |
| n | A long number of seconds |

**Return value:** Time. Returns the time that occurs n seconds after time if n is greater than 0. Returns the time that occurs n seconds before time if n is less than 0. The maximum return value is 23:59:59.

## Second

**Description:** Obtains the number of seconds in the seconds portion of a time value.

**Syntax:** Second ( time )

| Argument | Description |
|---|---|
| Time | The time value from which you want the seconds |

**Return value:** Integer. Returns the seconds portion of time (00 to 59).

## SecondsAfter

**Description:** Determines the number of seconds one time occurs after another.

**Syntax:** SecondsAfter ( time1, time2 )

| Argument | Description |
|---|---|
| Time1 | A time value that is the start time of the interval being measured |
| Time2 | A time value that is the end time of the interval |

**Return value:** Long. Returns the number of seconds time2 occurs after time1. If time2 occurs before time1, SecondsAfter returns a negative number.

## Time

**Description:** Converts a string to a time data type.

**Syntax:** Time ( string )

| Argument | Description |
|---|---|
| string | A string containing a valid time (such as 8AM or 10:25) that you want returned as a time data type. Only the hour is required; you do not have to include the minutes, |

| | |
|---|---|
| | seconds, or microseconds of the time or AM or PM. The default value for minutes and seconds is 00 and for microseconds is 000000. AM or PM is determined automatically |

**Return value:** Time. Returns the time in string as a time data type. If string does not contain a valid time, Time returns 00:00:00.

## Today

**Description:** Obtains the system date and time.

**Syntax:** Today ( )

**Return value:** DateTime. Returns the current system date and time.

## Year

**Description:** Determines the year of a date value.

**Syntax:** Year ( date )

| Argument | Description |
|---|---|
| date | The date from which you want the year |

**Return value:** Integer. Returns an integer whose value is a four-digit year adapted from the year portion of date if it succeeds and 1900 if an error occurs.
If the year is two digits, then program chooses the century, as follows. If the year is between 00 to 49, program assumes 20 as the first two digits; if it is between 50 and 99, program assumes 19.

**Usage:** Obtains the year portion of date. 24x7 Scheduler handles years from 1000 to 3000 inclusive. If your data includes dates before 1950, such as birth dates, always specify a 4-digit year so that Year function (and other functions) interprets the date as intended.

## Miscellanious functions

Case
If

ProfileInt
ProfileString
RGB

# Case

**Description:** Tests the values of a column or expression and returns values based on the results of the test.

**Syntax:**
Case ( column WHEN value1 THEN result1 { WHEN value2 THEN result2 { ...} } { ELSE resultelse } )

| Argument | Description |
|---|---|
| Column | The column or expression whose values you want to test. Column can be the column name or the column number preceded by a pound sign (#). Column can also be an expression that includes a reference to the column. Column is compared to each valuen |
| WHEN | Introduces a value-result pair. At least one WHEN is required |
| Valuen | One or more values that you want to compare to values of column. A value can be:<br>• A single value<br>• A list of values separated by commas (for example, 2, 4, 6, 8)<br>• A TO clause (for example, 1 TO 20)<br>• IS followed by a relational operator and comparison value (for example, IS>5)<br>• Any combination of the above with an implied OR between expressions (for example, 1,3,5,7,9,27 TO 33, IS>42) |
| THEN | Introduces the result to be returned when column matches the corresponding valuen |
| Resultn | An expression whose value is returned by Case for the corresponding valuen. All resultn must have the same data type |
| ELSE (optional) | (optional) Specifies that for any values of column that don't match the values of valuen already specified, Case returns resultelse |
| Resultelse | An expression whose value is returned by Case when the value of column doesn't match any WHEN valuen expression |

**Return value:** The data type of resultn. Returns the result you specify in resultn.

**Usage:** If more than one WHEN clause matches column, Case returns the result of the first matching one.

**Note:**
Function Case is similar to the ORACLE's Decode function.

## GetRow

**Description:** Reports the number of a row in which processing is performed.

**Syntax:** GetRow ( )

**Return value:** Long. Returns the number of a row if it succeeds, 0 if no data has been retrieved or added, and -1 if an error occurs.

## If

**Description:** Evaluates a condition and returns a value based on that condition.

**Syntax:** If ( boolean, truevalue, falsevalue )

| Argument | Description |
|---|---|
| boolean | A boolean expression that evaluates to TRUE or FALSE |
| Truevalue | A string containing the value you want returned if the boolean expression is TRUE |
| Falsevalue | A string containing the value you want returned if the boolean expression is FALSE |

**Return value:** The data type of truevalue or falsevalue. Returns truevalue if boolean is TRUE and falsevalue if it is FALSE. Returns NULL if an error occurs.

## IsRowModified

**Description:** Reports whether the row has been modified.

**Syntax:** IsRowModified ( )

**Return value:** Boolean. Returns TRUE if the row has been modified and FALSE if it has not.

Example: To filter out only modified rows, set filter expression as following:
IsRowModified ( )

# IsRowNew

**Description:** Reports whether the row has been newly inserted after last save or retrieve.

**Syntax:** IsRowNew ( )

**Return value:** Returns TRUE if the row is new and FALSE if it was retrieved from the database.

Example: To filter out only newly inserted rows, set filter expression as following:
IsRowNew ( )

**See also:**

Functions by category

# ProfileInt

**Description:** Obtains the integer value of a setting in the specified profile file.

**Syntax:** ProfileInt ( filename, section, key, default )

| Argument | Description |
|---|---|
| Filename | A string whose value is the name of the profile file. If you do not specify a full path, ProfileInt uses the operating system's standard file search order to find the file |
| section | A string whose value is the name of a group of related values in the profile file. In the file, section names are in square brackets. Do not include the brackets in section. Section is not case-sensitive |
| Key | A string whose value is the name of a group of related values in the profile file. In the file, section names are in square brackets. Do not include the brackets in section. Section is not case-sensitive |
| default | An integer value that ProfileInt will return if filename is not found, if section or key does not exist in filename, or if the value of key cannot be converted to an integer |

**Return value:** Integer. Returns default if filename is not found, section is not found in filename, or key is not found in section, or the value of key is not an integer. Returns -1 if an error occurs.

**Usage:** Use ProfileInt or ProfileString to get configuration settings from a profile file.

# ProfileString

**Description:** Obtains the string value of a setting in the specified profile file.

**Syntax:** ProfileInt ( filename, section, key, default )

| Argument | Description |
|---|---|
| Filename | A string whose value is the name of the profile file. If you do not specify a full path, ProfileString uses the operating system's standard file search order to find the file |
| Section | A string whose value is the name of a group of related values in the profile file. In the file, section names are in square brackets. Do not include the brackets in section. Section is not case-sensitive |
| Key | A string whose value is the name of a group of related values in the profile file. In the file, section names are in square brackets. Do not include the brackets in section. Section is not case-sensitive |
| default | An string value that ProfileString will return if filename is not found, if section or key does not exist in filename |

**Return value:** String, with a maximum length of 4096 characters. Returns the string from key within section within filename. If filename is not found, section is not found in filename, or key is not found in section, ProfileString returns default. If an error occurs, it returns the empty string ("").

**Usage:** Use ProfileInt or ProfileString to get configuration settings from a profile file.

# RGB

**Description:** Calculates the long value that represents the color specified by numeric values for the red, green, and blue components of the color.

**Syntax:** RGB ( red, green, blue )

| Argument | Description |
|---|---|
| Red | The integer value of the red component of the desired color |
| green | The integer value of the green component of the desired color |
| blue | The integer value of the blue component of the desired color |

**Return value:** Long. Returns the long that represents the color created by combining the values specified in red, green, and blue. If an error occurs, RGB returns NULL.

**Usage:** The formula for combining the colors is:

Red + (256 * Green) + (65536 * Blue)

Use RGB to obtain the long value required to set the color for text and drawing objects. You can also set an object's color to the long value that represents the color. The RGB function provides an easy way to calculate that value. The value of a component color is an integer between 0 and 255

# Numeric functions

Abs
Ceiling
Cos
Exp
Fact
Int
Log
LogTen
Mod
Pi
Rand
Round
Sign
Sin
Sqrt
Tan
Truncate

# Abs

**Description:** Calculates the absolute value of a number.

**Syntax:** Abs ( n )

| Argument | Description |
|---|---|
| n | The number for which you want the absolute value |

**Return value:** The data type of n. Returns the absolute value of n.

## Ceiling

**Description:** Determines the smallest whole number that is greater than or equal to a specified limit.

**Syntax:** Ceiling ( n )

| Argument | Description |
|---|---|
| N | The number for which you want the smallest whole number that is greater than or equal to it |

**Return value:** The data type of n. Returns the smallest whole number that is greater than or equal to n.

## Cos

**Description:** Calculates the cosine of an angle.

**Syntax:** Cos ( n )

| Argument | Description |
|---|---|
| n | The angle (in radians) for which you want the cosine |

**Return value:** Double. Returns the cosine of n.

## Exp

**Description:** Raises e to the specified power.

**Syntax:** Exp ( n )

| Argument | Description |
|---|---|
| n | The power to which you want to raise e (2.71828) |

**Return value:** Double. Returns e raised to the power n.

## Fact

**Description:** Determines the factorial of a number.

**Syntax:** Fact ( n )

| Argument | Description |
|---|---|
| n | The number for which you want the factorial |

**Return value:** Double. Returns the factorial of n.

## Int

**Description:** Determines the largest whole number less than or equal to a number.

**Syntax:** Int ( n )

| Argument | Description |
|---|---|
| n | The number for which you want the largest whole number that is less than or equal to it |

**Return value:** The data type of n. Returns the largest whole number less than or equal to n.

## Integer

**Description:** Converts the value of a string to an integer.

**Syntax:** Integer ( string )

| Argument | Description |
|---|---|
| string | The string you want returned as an integer |

**Return value:** Integer. Returns the contents of string as an integer if it succeeds and 0 if string is not a number.

# Log

**Description:** Determines the natural logarithm of a number.

**Syntax:** Log ( n )

| Argument | Description |
|---|---|
| n | The number for which you want the natural logarithm (base e). The value of n must be greater than 0 |

**Return value:** Double. Returns the natural logarithm of n. An execution error occurs if n is negative or zero.

**Inverse:** The inverse of the Log function is the Exp function.

# LogTen

**Description:** Determines the base 10 logarithm of a number.

**Syntax:** LogTen ( n )

| Argument | Description |
|---|---|
| n | The number for which you want the base 10 logarithm. The value of n must not be negative |

**Return value:** Double. Returns the decimal log of n.

**Obtaining a number:** The expression 10^n is the inverse for LogTen(n). To obtain n given number (nbr = LogTen(n)), use n = 10^r.

# Long

**Description:** Converts the value of a string to a long.

**Syntax:** Long ( string )

| Argument | Description |
|---|---|
| string | The string you want returned as a long |

**Return value:** Long. Returns the contents of string as a long if it succeeds and 0 if string is not a valid number.

## Mod

**Description:** Obtains the remainder (modulus) of a division operation.

**Syntax:** Mod ( x, y )

| Argument | Description |
|---|---|
| X | The number you want to divide by y |
| Y | The number you want to divide into x |

**Return value:** The data type of x or y, whichever data type is more precise.

## Number

**Description:** Converts a string to a number.

**Syntax:** Number ( string )

| Argument | Description |
|---|---|
| string | The string you want returned as a number |

**Return value:** A numeric data type. Returns the contents of string as a number. If string is not a valid number, Number returns 0.

## Pi

**Description:** Multiplies pi by a specified number.

**Syntax:** Pi ( n )

| Argument | Description |
|---|---|
| n | The number you want to multiply by pi (3.14159265358979323...) |

**Return value:** Double. Returns the result of multiplying n by pi if it succeeds and -1 if an error occurs.

**Usage:** Use Pi to convert angles to and from radians.

## Rand

**Description:** Obtains a random whole number between 1 and a specified upper limit.

**Syntax:** Rand ( n )

| Argument | Description |
|---|---|
| n | The upper limit of the range of random numbers you want returned. The lower limit is always 1. The upper limit cannot exceed 32,767 |

**Return value:** A numeric data type, the data type of n. Returns a random whole number between 1 and n.

**Usage:** The sequence of numbers generated by repeated calls to the Rand function is a computer-generated pseudo-random sequence.

## Round

**Description:** Rounds a number to the specified number of decimal places.

**Syntax:** Round ( x, n )

| Argument | Description |
|---|---|
| X | The number you want to round |
| n | The number of decimal places to which you want to round x |

**Return value:** Decimal. If n is positive, returns x rounded to the specified number of decimal places. If n is negative, returns x rounded to (-n+1) places before the decimal point. Returns -1 if it fails.

## Sign

**Description:** Determines the sign of a number. Sign reports whether a number is negative, zero, or positive.

**Syntax:** Sign ( n )

| Argument | Description |
|---|---|
| n | The number for which you want to determine the sign |

**Return value:** Integer. Returns a number (-1, 0, or 1) indicating the sign of n.

## Sin

**Description:** Calculates the sine of an angle.

**Syntax:** Sin ( n )

| Argument | Description |
|---|---|
| n | The angle (in radians) for which you want the sine |

**Return value:** Double. Returns the sine of n.

## Sqrt

**Description:** Calculates the square root of a number.

**Syntax:** Sqrt ( n )

| Argument | Description |
|---|---|
| n | The number for which you want the square root |

**Return value:** Double. Returns the square root of n.

**Usage:** Sqrt(n) is the same as n^.5.
Taking the square root of a negative number causes an execution error.

## Tan

**Description:** Calculates the tangent of an angle.

**Syntax:** Tan ( n )

| Argument | Description |
|---|---|
| n | The angle (in radians) for which you want the tangent |

**Return value:** Double. Returns the tangent of n if it succeeds and -1 if an error occurs.

## Truncate

**Description:** Truncates a number to the specified number of decimal places.

**Syntax:** Truncate ( x, n )

| Argument | Description |
|---|---|
| X | The number you want to truncate |
| n | The number of decimal places to which you want to truncate x |

**Return value:** The data type of n. If n is positive, returns x truncated to the specified number of decimal places. If n is negative, returns x truncated to (-n+1) places before the decimal point. Returns -1 if it fails.

## String functions

Fill
Left
LeftTrim
Len
Lower
Match
Mid
Pos
Replace
Right
RightTrim
Space

String
Trim
Upper
WordCap

## Asc

**Description:** Converts the first character of a string to its ASCII integer value.

**Syntax:** Asc ( string )

| Argument | Description |
|---|---|
| String | The string for which you want the ASCII value of the first character |

**Return value:** Integer. Returns the ASCII value of the first character in string.

**Usage:** Use Asc to test the case of a character or manipulate text and letters.To find out the case of a character, you can check whether its ASCII value is within the appropriate range.

## Char

**Description:** Converts an integer to a character.

**Syntax:** Char ( n )

| Argument | Description |
|---|---|
| n | The integer you want to convert to a character |

**Return value:** String. Returns the character whose ASCII value is n.

## Fill

**Description:** Builds a string of the specified length by repeating the specified characters until the result string is long enough.

**Syntax:** Fill ( chars, n )

| Argument | Description |
|---|---|
| Chars | A string whose value will be repeated to fill the return string |
| n | A long whose value is the length of the string you want returned (from 0 to 60000) |

**Return value:** String. Returns a string n characters long filled with repetitions of the characters in the argument chars. If the argument chars has more than n characters, the first n characters of chars are used to fill the return string. If the argument chars has fewer than n characters, the characters in chars are repeated until the return string has n characters.

## Left

**Description:** Obtains a specified number of characters from the beginning of a string.

**Syntax:** Left ( string, n )

| Argument | Description |
|---|---|
| string | The string containing the characters you want |
| n | A long specifying the number of characters you want |

**Return value:** String. Returns the leftmost n characters in string if it succeeds and the empty string ("") if an error occurs. If n is greater than or equal to the length of the string, Left returns the entire string. It does not add spaces to make the return value's length equal to n.

## LeftTrim

**Description:** Removes leading spaces from the beginning of a string.

**Syntax:** LeftTrim ( string )

| Argument | Description |
|---|---|
| string | The string you want returned with leading blanks deleted |

**Return value:** String. Returns a copy of string with leading blanks deleted if it succeeds and the empty string ("") if an error occurs.

## Len

**Description:** Reports the length of a string.

**Syntax:** Len ( string )

| Argument | Description |
|---|---|
| string | The string for which you want the length |

**Return value:** Long. Returns a long containing the length of string if it succeeds and -1 if an error occurs.

## Lower

**Description:** Converts all the characters in a string to lowercase.

**Syntax:** Lower ( string )

| Argument | Description |
|---|---|
| string | The string you want to convert to lowercase letters |

**Return value:** String. Returns string with uppercase letters changed to lowercase if it succeeds and the empty string ("") if an error occurs.

## Match

**Description:** Determines whether a string's value contains a particular pattern of characters.

**Syntax:** Match ( string, textpattern )

| Argument | Description |
|---|---|
| string | The string in which you want to look for a pattern of characters |
| Textpattern | A string whose value is the text pattern |

**Return value:** Boolean. Returns TRUE if string matches textpattern and FALSE if it does not. Match also returns FALSE if either argument has not been assigned a value or the pattern is invalid.

**Usage:** Match enables you to evaluate whether a string contains a general pattern of characters. To find out whether a string contains a specific substring, use the Pos function.
Textpattern is similar to a regular expression. It consists of metacharacters, which have special meaning, and ordinary characters, which match themselves. You can specify that the string begin or end with one or more characters from a set, or that it contains any characters except those in a set.

A text pattern consists of **metacharacters**, which have special meaning in the match string, and **non-metacharacters**, which match the characters themselves.

**See also:**

Metacharacters and sample patterns

## Metacharacters

The following tables explain the meaning and use of the metacharacters:

| Metacharacter | Meaning | Example |
|---|---|---|
| Caret (^) | Matches the beginning of a string | ^C matches C at the beginning of a string |
| Dollar sign ($) | Matches the end of a string | s$ matches s at the end of a string |
| Period (.) | Matches any character | . . . matches three consecutive characters |
| Backslash (\) | Removes the following metacharacter's special characteristics so that it matches itself | \$ matches $ |
| Character class (a group of characters enclosed in square brackets ([ ])) | Matches any of the enclosed characters | [AEIOU] matches A, E, I, O, or Uyou can use hyphens to abbreviate ranges of characters in a character class. For example, [A-Za-z] matches any letter |
| Complemented character class (first character inside the brackets is a caret) | Matches any character not in the group following the caret | [^0-9] matches any character except a digit, and [^A-Za-z] matches any character except a letter |

The metacharacters asterisk (*), plus (+), and question mark (?) are unary operators that are used to specify repetitions in a regular expression:

| Metacharacter | Meaning | Example |
|---|---|---|
| * (asterisk) | Indicates zero or more occurrences | A* matches zero or more As (no As, A, AA, AAA, and so on) |
| + (plus) | Indicates one or more occurrences | A+ matches one A or more than one A (A, AAA, and so on) |
| ? (question mark) | Indicates zero or one occurrence | A? matches an empty string ("") or A |

### Sample patterns

The following table shows various text patterns and sample text that matches each pattern:

| This pattern | Matches |
|---|---|
| AB | Any string that contains AB; for example, ABA, DEABC, graphAB_one |
| B* | Any string that contains 0 or more Bs; for example, AC, B, BB, BBB, ABBBC, and so on |
| AB*C | Any string containing the pattern AC or ABC or ABBC, and so on (0 or more Bs) |
| AB+C | Any string containing the pattern ABC or ABBC or ABBBC, and so on (1 or more Bs) |
| ABB*C | Any string containing the pattern ABC or ABBC or ABBBC, and so on (1 B plus 0 or more Bs) |
| ^AB | Any string starting with AB |
| AB?C | Any string containing the pattern AC or ABC (0 or 1 B) |
| ^[ABC] | Any string starting with A, B, or C |
| [^ABC] | A string containing any characters other than A, B, or C |
| ^[^abc] | A string that begins with any character except a, b, or c |
| ^[^a-z]$ | Any single-character string that is not a lowercase letter (^ and $ indicate the beginning and end of the string) |
| [A-Z]+ | Any string with one or more uppercase letters |
| ^[0-9]+$ | Any string consisting only of digits |
| ^[0-9][0-9][0-9]$ | Any string consisting of exactly three digits |
| ^([0-9][0-9][0-9])$ | Any consisting of exactly three digits enclosed in parentheses |

**See also:**

Match function

## Mid

**Description:** Obtains a specified number of characters from a specified position in a string.

**Syntax:** Mid ( string, start {, length } )

| Argument | Description |
|---|---|
| string | The string from which you want characters returned |
| start | A long specifying the position of the first character you |

| | |
|---|---|
| | want returned. (The position of the first character of the string is 1) |
| Length (optional) | A long whose value is the number of characters you want returned. If you do not enter length or if length is greater than the number of characters to the right of start, Mid returns the remaining characters in the string |

**Return value:** String. Returns characters specified in length of string starting at character start. If start is greater than the number of characters in string, the Mid function returns the empty string (""). If length is greater than the number of characters remaining after the start character, Mid returns the remaining characters. The return string is not filled with spaces to make it the specified length.

## Pos

**Description:** Finds one string within another string.

**Syntax:** Pos ( string1, string2 {, start } )

| Argument | Description |
|---|---|
| string1 | The string in which you want to find string2 |
| string2 | The string you want to find in string1 |
| start (optional) | A long indicating where the search will begin in string1. The default is 1 |

**Return value:** Long. Returns a long whose value is the starting position of the first occurrence of string2 in string1 after the position specified in start. If string2 is not found in string1 or if start is not within string1, Pos returns 0.

**Usage:** The Pos function is case sensitive.

## Replace

**Description:** Replaces a portion of one string with another.

**Syntax:** Replace ( string1, start, n, string2 )

| Argument | Description |
|---|---|
| string1 | The string in which you want to replace characters with string2 |
| start | A long whose value is the number of the first character you want replaced. (The first character in the string is number 1) |

| | |
|---|---|
| n | A long whose value is the number of characters you want to replace |
| string2 | The string that will replace characters in string1. The number of characters in string2 can be greater than, equal to, or fewer than the number of characters you are replacing |

**Return value:** String. Returns the string with the characters replaced if it succeeds and the empty string ("") if it fails.

**Usage:** If the start position is beyond the end of the string1, Replace appends string2 to string1. If there are fewer characters after the start position than specified in n, Replace replaces all the characters to the right of character start.
If n is zero, then in effect Replace inserts string2 into string1.

## Right

**Description:** Obtains a specified number of characters from the end of a string.

**Syntax:** Right ( string, n )

| Argument | Description |
|---|---|
| string | The string containing the characters you want |
| n | A long specifying the number of characters you want |

**Return value:** String. Returns the rightmost n characters in string if it succeeds and the empty string ("") if an error occurs. If n is greater than or equal to the length of the string, Right returns the entire string. It does not add spaces to make the return value's length equal to n.

## RightTrim

**Description:** Removes spaces from the end of a string.

**Syntax:** RightTrim ( string )

| Argument | Description |
|---|---|
| string | The string you want returned with trailing blanks deleted |

**Return value:** String. Returns a copy of string with trailing blanks deleted if it succeeds and the empty string ("") if an error occurs.

# Space

**Description:** Builds a string of the specified length whose value consists of spaces.

**Syntax:** Space ( n )

| Argument | Description |
|---|---|
| n | A long whose value is the length of the string you want filled with spaces |

**Return value:** String. Returns a string filled with n spaces if it succeeds and the empty string ("") if an error occurs.

# String

**Description:** Formats data as a string according to a specified format mask. You can convert and format date, DateTime, numeric, and time data. You can also apply a format to a string.

**Syntax:** String ( data {, format } )

| Argument | Description |
|---|---|
| Data | The data you want returned as a string with the specified formatting. Data can have a date, DateTime, numeric, time, or string data type |
| Format (optional) | A string of the masks you want to use to format the data. The masks consist of formatting information specific to the data type of data. If data is type string, format is required. The format string can consist of more than one mask, depending on the data type of data. Each mask is separated by a semicolon. See Usage for details on each data type |

**Return value:** String. Returns data in the specified format if it succeeds and the empty string ("") if the data type of data does not match the type of mask specified or format is not a valid mask.

**Usage:** For date, DateTime, numeric, and time data, program uses the system's default format for the returned string if you don't specify a format. For numeric data, the default format is the [General] format. For string data, a format mask is required. (Otherwise, the function would have nothing to do.) The format can consist of one or more masks:

- Formats for date, DateTime, string, and time data can include one or two masks. The first mask is the format for the data; the second mask is the format for a null value.
- Formats for numeric data can have up to four masks. A format with a single mask handles both positive and negative data. If there are additional masks, the first mask is for positive values, and the additional masks are for negative, zero, and NULL values.

If the format doesn't match the data type, 24x7 Scheduler will try to apply the mask, which can produce unpredictable results.

**See also:**

Format symbols

## Format Symbols

The following table lists the format symbols that can be used in a format string:

| Format Symbol | Description |
|---|---|
| [General] | Outputs the number in General format. |
| 0 | Digit placeholder. If the number contains fewer digits than the format contains placeholders, the number is padded with 0's. If there are more digits to the right of the decimal than there are placeholders, the decimal portion is rounded to the number of places specified by the placeholders. If there are more digits to the left of the decimal than there are placeholders, the extra digits are retained. |
| # | Digit placeholder. This placeholder functions the same as the 0 placeholder except the number is not padded with 0's if the number contains fewer digits than the format contains placeholders. |
| ? | Digit placeholder. This placeholder functions the same as the 0 placeholder except that spaces are used to pad the digits. |
| . (period) | Decimal point. Determines how many digits (0's or #'s) are output on either side of the decimal point. If the format contains only #'s left of the decimal point, numbers less than 1 begin with a decimal point. If the format contains 0's left of the decimal point, numbers less than 1 begin with a 0 left of the decimal point. |
| % | Outputs the number as a percentage. The number is multiplied by 100 and the % character is appended. |
| , (comma) | Thousands separator. If the format contains commas separated by #'s or 0's, the number is output with commas separating thousands. A comma following a placeholder scales the number by a thousand. For Example, the format 0, scales the number by 1000 (e.g., 10,000 would be converted as 10). |
| E- E+ e- e+ | Outputs the number as scientific notation. If the format contains a scientific notation symbol to the left of a 0 or # placeholder, the number is output in scientific notation and an E or an e is added. The number of 0 and # placeholders to the right of the decimal determines the number of digits in the exponent. E- and e- place a minus sign by negative exponents. E+ and e+ place a minus sign by negative exponents and a plus sign by positive exponents. |
| $ - + / ( ) : space | Outputs that character. To output a character other than those listed, precede the character with a back slash (\) or enclose the character in double quotation marks (""). You can also use the slash (/) for fraction formats. |
| \ | Outputs the next character. The backslash is not output. You can also output a character or string of characters by surrounding the characters with double quotation marks (""). |
| * (asterisk) | Repeats the next character until the width of the column is filled. You cannot have more than one asterisk in each format section. |
| [ ] | Outputs hours greater than 24, or minutes or seconds greater than 60. Place the brackets around the leftmost part of the time code; for Example, [h]:mm:ss would allow the output of hours greater than 24. |
| "text" | Outputs the text inside the quotation marks. |

| | |
|---|---|
| @ | Text placeholder. Text replaces the @ format character. |
| m | Month number. Outputs the month as digits without leading zeros (e.g., 1-12). Can also represent minutes when used with h or hh formats. |
| mm | Month number. Outputs the month as digits with leading zeros (e.g., 01-12). Can also represent minutes when used with the h or hh formats. |
| mmm | Month abbreviation. Outputs the month as an abbreviation (e.g., Jan-Dec). |
| mmmm | Month name. Outputs the month as a full name (e.g., January-December). |
| d | Day number. Outputs the day as digits with no leading zero (e.g., 1-9). |
| dd | Day number. Outputs the day as digits with leading zeros (e.g., 01-31). |
| ddd | Day abbreviation. Outputs the day as an abbreviation (e.g., Sun-Sat). |
| dddd | Day name. Outputs the day as a full name (e.g., Sunday-Saturday). |
| yy | Year number. Outputs the year as a two-digit number (e.g., 00-99). |
| yyyy | Year number. Outputs the year as a four-digit number (e.g., 1900-2078). |
| h | Hour number. Outputs the hour as a number without leading zeros (1-23). If the format contains one of the AM or PM formats, the hour is based on a 12-hour clock. Otherwise, it is based on a 24-hour clock. |
| hh | Hour number. Outputs the hour as a number with leading zeros (01-23). If the format contains one of the AM or PM formats, the hour is based on a 12-hour clock. Otherwise, it is based on a 24-hour clock. |
| m | Minute number. Outputs the minute as a number without leading zeros (0-59). The m format must appear immediately after the h or hh symbol. Otherwise, it is interpreted as a month number. |
| mm | Minute number. Outputs the minute as a number with leading zeros (00-59). The mm format must appear immediately after the h or hh symbol. Otherwise, it is interpreted as a month number. |
| s | Second number. Outputs the second as a number without leading zeros (0-59). |
| ss | Second number. Outputs the second as a number with leading zeros (00-59). |
| AM/PM<br>am/pm<br>A/P<br>a/p | 12-hour time. Outputs time using a 12-hour clock. Outputs AM, am, A, or a for times between midnight and noon; outputs PM, pm, P, or p for times from noon until midnight. |

# Trim

**Description:** Removes leading and trailing spaces from a string.

**Syntax:** Trim ( string )

| Argument | Description |
|---|---|
| string | The string you want returned with leading and trailing |

| | |
|---|---|
| | spaces deleted |

**Return value:** String. Returns a copy of string with all leading and trailing spaces deleted if it succeeds and the empty string ("") if an error occurs.

## Upper

**Description:** Converts all the characters in a string to uppercase.

**Syntax:** Upper ( string )

| Argument | Description |
|---|---|
| String | The string you want to convert to uppercase letters |

**Return value:** String. Returns string with lowercase letters changed to uppercase if it succeeds and the empty string ("") if an error occurs.

## WordCap

**Description:** Sets the first letter of each word in a string to a capital letter and all other letters to lowercase (for example, ROBERT E. LEE would be Robert E. Lee).

**Syntax:** WordCap ( string )

| Argument | Description |
|---|---|
| string | A string or expression that evaluates to a string with initial capital letters (for example, Monday Morning) |

**Return value:** String. Returns string with the first letter of each word set to uppercase and the remaining letters lowercase if it succeeds and NULL if an error occurs.

## Operators

An operator is a symbol or word in an expression that performs an arithmetic calculation or logical operation; compares numbers, text, or values; or manipulates text strings.

The database buffer supports the following types of operators:
- Arithmetic for numeric data types
- Relational for all data types
- Logical for all data types
- Concatenation for string data types

**Arithmetic operators**

These are the arithmetic operators:

| Operator | Meaning | Example |
|---|---|---|
| + | Addition | SubTotal + Tax |
| - | Subtraction | Price - Discount |
| * | Multiplication | Quantity * Price |
| / | Division | Discount / Price |
| ^ | Exponentiation | Rating ^ 2.5 |

**Relational and logical operators**

Logical operators can join relational expressions to form more complex boolean expressions.
The result of evaluating a boolean expression is always TRUE or FALSE.
These are the relational and logical operators:

| Operator | Meaning | Example |
|---|---|---|
| = | Equals | Price = 100 |
| > | Greater than | Price > 100 |
| < | Less than | Price < 100 |
| <> | Not equal | Price <> 100 |
| >= | Greater than or equal | Price >= 100 |
| <= | Less than or equal | Price <= 100 |
| NOT | Logical negation | NOT Price = 100 |
| AND | Logical and | Tax > 3 AND Ship < 5 |
| OR | Logical or | Tax >3 OR Ship < 5 |
| IN | SQL SELECT IN condition | Status IN ('A', 'S', 'R') |
| BETWEEN | SQL SELECT BETWEEN condition | Ship BETWEEN 3 AND 5 |
| LIKE | SQL SELECT LIKE condition | Brand LIKE 'Micro%' |

**Concatenation operator**

The concatenation operator joins the contents of two string expressions to form a longer value. You can concatenate strings only.

| Operator | Meaning | Example |
|---|---|---|
| + | Concatenate | "cat "+ "dog" |

**Operator precedence in expressions**

To ensure predictable results, all operators in an expression are evaluated in a specific order of precedence. When the operators have the same precedence, the database buffer evaluates them left to right.

The following table lists the operators in descending order of precedence:

| **Operator** | **Purpose** |
|---|---|
| ( ) | Grouping |
| ^ | Exponentiation |
| *, / | Multiplication and division |
| +, - | Addition and subtraction; string concatenation |
| IN, LIKE, BETWEEN | SQL SELECT statement conditions |
| NOT | Logical negation |
| =, >, <, <=, >=, <> | Relational operators |
| AND, OR | Logical AND and logical OR |

# Chapter 20: Visual Basic Script

## VBScript Overview

Microsoft® Visual Basic® Scripting Edition, a subset of the Microsoft® Visual Basic® programming language, can be used for advanced automation with 24x7 Scheduler. The primary purpose for it in 24x7 Scheduler is to provide automation support for ActiveX Controls (formerly OLE), and various Automation servers.

**See also:**

Scripting Conventions
VBScript Language Reference
Cross-Language Calls

## Scripting Conventions

The code for a Visual Basic Script Job or notification action must consist of one or more subroutines and functions. You use the **Sub** statement to declare subroutines and **Function** statement to declare functions. Every job and notification action script written in VBScript must contain the Main subroutine. The 24x7 Scheduler uses this subroutine as an entry point for your script. If the Main subroutine cannot be found in run-time, an error will occur.

From the Main subroutine you can call other subroutines and functions using valid Visual Basic Script syntax and calling conventions. Do not write VBScript code outside subroutines and functions as it may cause unpredictable results and will not be supported in future versions of 24x7 Scheduler.

You can use **macro-parameters** within your VBScript code just as you use them in other places.

**Examples of valid scripts:**

*Example 1:*

```
' This script shows "Hello" message
Sub Main()
   MsgBox "Hello"
End Sub
```

*Example 2:*

```
' This script shows "Hello" message using a call to another subroutine
Sub Main()
   Call ShowMessage( "Hello")
End Sub


Sub ShowMessage( argMessage )
   MsgBox argMessage
End Sub
```

*Example 3:*

```
' This script uses VBScript built-in FileSystemObject to find out the last
' time a file was modified
```

```
Sub Main()
   Dim fso, f1
   Set fso = CreateObject("Scripting.FileSystemObject")
   ' Get a File object to query.
   Set f1 = fso.GetFile("c:\detlog.txt")
   MsgBox "File last modified: " & f1.DateLastModified

   ' Put this time into a global variable
   JALScript.Execute("Dim GLOBAL.logtime, datetime" & vbCrLf & _
               "Set GLOBAL.logtime, """ & f1.DateLastModified & """")
End Sub
```

# VBScript Extensions

24x7 Scheduler provides **vb24x7** COM object for extending functionality of the standard VBScript collection of functions and methods and simplifying automation projects. At present time this **vb24x7** COM object group of methods for starting and manipulating processes and printing files.

Here is an example of how this object can be used

*Example:*

```
' This script starts Notepad and then after 5 seconds automatically
' terminates it.

Sub Main()
   Dim vb24x7
   Set vb24x7 = CreateObject("vb24x7.Process")
   ProcessId = vb24x7.RunAndWait("Notepad.exe", 50)
   If ProcessId < 0 Then MsgBox vb24x7.LastError
   Set vb24x7 = Nothing
End Sub
```

vb24x7.Process class supports the following methods:

**Run**
**RunAndWait**
**KillProcess**
**PrintFile**
**RunJob**

# VBScript Run method

**Description:** Runs the specified program or document. This function is available in **vb24x7.Process** COM object.

**Syntax:** Function Run(CommandLine As String) As Long

| Argument | Description |
|---|---|
| CommandLine | A string whose value is the full or partial path and filename of the module to execute.<br>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.<br><br>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:<br><br>13 The directory from which the 24x7 Scheduler loaded.<br>14 The current directory.<br>15 The 32-bit Windows system directory. The name of this directory is SYSTEM32.<br>16 The 16-bit Windows system directory. The name of this directory is SYSTEM.<br>17 The Windows directory. The name of this directory is WINDOWS or WINNT.<br>18 The directories that are listed in the PATH environment variable.<br><br>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.<br><br>Program name can be followed by **command line parameters**. You can use **macro-parameters** and **system environment variables** within program name and command line parameters string to pass dynamic information (such as the current month) to the scheduled program. |

**Return value:** Number. Returns ID of the created process or returns a negative number if an error occurs. You can use the returned ID to control and manipulate further process execution.

**Usage:** You can use Run for any application that you can run from the operating system. If you specify command line parameters, the application determines the meaning of those parameters. A typical use for command line parameters is to identify a data file to be opened when the program is executed.
In order to run a document, its extension must be registered. For example, if you want to start a MDB file that has AutoExec macro, you must have MDB file extension registered as a MS Access database application.

**See also:**
- RunAndWait
- KillProcess
- RunJob

# VBScript RunAndWait method

**Description:** Starts specified program or document and enters an efficient wait state until this process finishes or the timeout interval elapses. In the latter case, the 24x7 Scheduler forcedly terminates the process. This function is available in **vb24x7.Process** COM object.

**Syntax:** Function Run(CommandLine As String, Timeout As Long) As Long

| Argument | Description |
|---|---|
| CommandLine | A string whose value is the full or partial path and filename of the module to execute.<br>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.<br><br>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:<br><br>19 The directory from which the 24x7 Scheduler loaded.<br>20 The current directory.<br>21 The 32-bit Windows system directory. The name of this directory is SYSTEM32.<br>22 The 16-bit Windows system directory. The name of this directory is SYSTEM.<br>23 The Windows directory. The name of this directory is WINDOWS or WINNT.<br>24 The directories that are listed in the PATH environment variable.<br><br>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.<br><br>Program name can be followed by **command line parameters**. You can use **macro-parameters** and **system environment variables** within program name and command line parameters string to pass dynamic information (such as the current month) to the scheduled program. |
| Timeout | A number whose value is the maximum time interval (in seconds) within which you allow the specified program to run. Use 0 timeout to allow infinite waiting. |

**Return value:** Number. Returns ID of the created process or a negative number if an error occurs.

**Usage:** You can use RunAndWait for any application that you can run from the operating system. If you specify command line parameters, the application determines the meaning of those parameters. A typical use for command line parameters is to identify a data file to be opened when the program is executed.
In order to run a document, its extension must be registered. So that if you want to start MDB files that has AutoExec macro you must have MDB file extension registered as a MS Access database application.
Use RunAndWait when you have a complex job with one or more program dependencies so the next program starts upon completion of the process specified by CommandParam.

**See also:**

Run
KillProcess
RunJob

# VBScript KillProcess method

**Description:** Terminates the specified process and all of its threads. This function is available in **vb24x7.Process** COM object.

**Syntax:** Function KillProcess(ProcessID As Long) As Long

| Argument | Description |
|---|---|
| ProcessID | A number whose value is the id of the process which you want to terminate |

**Return value:** Number. Returns 1 if success or returns a negative number if an error occurs.

**Usage:** The KillProcess function can be used to unconditionally cause a process to exit.

**{bmct note.bmp} Note:**
Termination of a process does not always forces termination of all its child processes!

**See also:**
Run
RunJob

# VBScript RunJob method

**Description:** Runs the specified job. This function is available in **vb24x7.Process** COM object.

**Syntax:** Function RunJob(JobID As String, Wait As Boolean, Timeout As Long) As Long

| Argument | Description |
|---|---|
| JobID | A string whose value is ether job ID or job name |
| Wait | A boolean whose value indicates whether the system should wait for the job to complete before returning control back to the calling script. |
| Timeout | A number whose value is the maximum time interval (in seconds) within which you allow the specified job to run. Use 0 timeout to allow infinite waiting.<br>This parameter is ignored if the Wait parameter is set to False |

**Return value:** None.

Use RunJob in your scripts to programmatically start other jobs. This allows creation of simple and complex batch jobs.

**See also:**
Run
RunAndWait

# VBScript PrintFile method

**Description:** Sends the specified file to the default printer. This function is available in **vb24x7.Process** COM object.

**Syntax:** Function PrintFile(FileName As String) As Long

| Argument | Description |
|---|---|
| FileName | A string whose value is the name of the file you want to print. If FileName is not on the operating system's search path, you must enter the fully qualified name. |

**Return value:** None.

**See also:**
PrinterSetDefault statement available in JAL

# Cross-Language Calls

24x7 Scheduler supports two different scripting environments: Job Automation Language (called briefly JAL) and Visual Basic Script (called VBScript or VBS). it also provides methods for these environments to communicate with each other.

In your JAL scripts you can use VBScriptExecute statement to dynamically execute scripts written in VBScript.

In your Visual Basic scripts you can use public object JALScript and its public Execute method to dynamically execute scripts written in JAL.

**JAL to VBScript calls –** VBScriptExecute

**Description:** Executes specified Visual Basic Script

**Syntax:** VBScriptExecute script

| Argument | Description |
|---|---|
| Script | A string whose value is the Visual Basic Script that you want to execute from your JAL script. |

**Return value:** None.

**Usage:** Use VBScriptExecute statement to execute methods available in VBScript and not available in JAL. For example, you can add COM automation to your existing JAL jobs.

**VBScript to JAL calls –** JALScript.Execute

**Description:** Executes specified JAL script

**Syntax:** JALScript.Execute script

| Argument | Description |
|---|---|
| Script | A string whose value is the JAL script that you want to execute from your Visual Basic script. |

**Return value:** None.

**Usage:** Use JALScript.Execute method to execute methods available in JAL script and not available in VBScript. For example, you can add FTP automation to your existing VBScript jobs.

# Chapter 21: Script Debugger

**Description**

Debugging is a process you use to find and resolve errors, or bugs, in your JAL scripts. There are three types of errors you may encounter as your script runs:

- Syntax errors as a result of incorrectly constructed code. You may have forgotten to balance pairs of statements (such as end labels for LoopWhile, LoopUntil, ForNext loops), or you may have a programming mistake that violates the rules of JAL (such as a misspelled word, missing separator, or argument type mismatch error, mismatched parentheses or an incorrect number of arguments passed to a JAL statement).
- Run-time specific errors after the job starts to run. Examples of run-time errors include attempting an illegal operation, such as dividing by zero or writing to a file that does not exist.
- Logic errors when the script does not perform as intended and produces incorrect results.

To help you isolate all three types of errors and to monitor how your code runs, 24x7 Scheduler provides debugging tools that let you step through your code one line at a time, examine or monitor the values of variables, and trace statement calls.

**Using the Debugger**

To use the debugger, set breakpoints in the scripts that you want to examine and click the Start button (or press the F5 key). When the script stops at a breakpoint, you can examine the values of variables and view the call stack. The debugger lets you set breakpoints as well as view and change local script and global variables. You can single-step through the code, continue to the next breakpoint, or skip a few lines and continue running from the cursor location.

**Breakpoints**

Breakpoint is a line of code in a job script or user-defined JAL statement at which the debugger automatically suspends execution.

To make the Debugger pause in its execution of your code, you can set a breakpoint:
- In the script text pane, double-click on the desired line of code that is not already a breakpoint.
  Or
- Move the caret to the desired line then either press the F8 key or select the **Toggle Breakpoint** command from the Debug menu.

To remove a breakpoint:
- Double-click the line of code on which the breakpoint has been set.
  Or
- Move the caret to the breakpoint line then either press the F8 key or select the **Toggle Breakpoint** command from the Debug menu.

A breakpoint can be set on a line that contains an executable statement

**Views in the Debugger window**

The debugger has two panes. You can resize these at any time.

The top pane shows the full text of a currently executed script. In that pane you can go to a specific line in a script, find a string, print the displayed script, and manage the breakpoints.

The bottom pane contains four tab pages:

| | |
|---|---|
| **Variables Watch -** | An expandable list of all the variables in scope. The values in the Variables Watch tab page are updated each time they are changed from a script or Immediate window. You can double-click on any variable to manually change the value of a variable whenever script execution is suspended. |
| **Call Stack -** | The sequence of JAL statement calls leading up to the script that was executing at the time of the breakpoint. |
| **Breakpoints -** | A list of breakpoints in the current script. |
| **Immediate -** | A part of the Debugger window in which you can run individual lines of JAL code, |

| | |
|---|---|
| | usually for the purpose of debugging. The Immediate pane is a kind of scratch pad window in which statements and variables are evaluated immediately. In the Immediate page you can type a statement then press F6 to execute this statement. |

**Step through code**

Stepping through your script can help you identify where an error is occurring. You can see whether each line of code produces the results you expected. To step through the code after you have it opened in the Debugger, do one of the following:

- To step through each line of job script and into the code in a user-defined statement called by the main script or by another statement, select the **Debug/Step** command from the Debug menu, or alternatively you can press the CTRL+F5 keys
- To run the code that precedes the desired line of code, and then break so you can step through each line of code, set a breakpoint on that line then select the **Start** or **Continue** command from the Debug menu. The Debugger will suspend execution at the breakpoint line. Use the **Debug/Step** command to continue line by line (see previous paragraph for details).
- To run the rest of the current script, select the **Continue** command from the Debug menu. The Debugger will continue executing script until it reaches a breakpoint or an end of script, whichever is first.
- To skip undesired lines, move the caret to the desired line, select the **Set Next Statement** command from the Debug menu. Select the **Step** or **Continue** command from the Debug window.

The type of stepping you do depends on which portions of code you want to analyze.

# Chapter 22: Testing and Debugging

Sometimes a 24x7 script type job does not behave the way you think it will. Perhaps a variable is not being assigned the value you expect, or a script does not do what you want it to do. In such situations, you can debug the script by using the following options:

1 Using the **Debugger,** run your script in debug mode. This allows you to monitor your script execution; evaluate and change variables, and trace the call stack.
2 Call the **MessageBox** statement within the script to display the value of a variable.
3 Call the **DatabaseDescribe** statement within the script to get definition of the database result set that was created when preparing for data retrieval.
4 Enable the **Trace** features then execute the job. The 24x7 Scheduler writes detailed tracing information into the SCRIPT.LOG file, which you can examine using the **Log Viewer**.
5 Enable the **Database Trace** features then execute the job. The 24x7 Scheduler writes detailed tracing information for all database calls into the PBTRACE.LOG file. You can examine this using the **Log Viewer**.
6 Use the **TelnetConfig** statement to show the Telnet terminal window, which displays all Telnet session messages, sent between the 24x7 Scheduler and the remote host.

After you have found and fixed the problems in your scripts, you should remove all MessageBox statements, so that the job can run without user intervention. You should also disable all tracing features to allow this and other jobs to run at optimum speed without performance hit caused by the tracing.

**See also:**

Troubleshooting Job Execution
Troubleshooting Database Connection
Execution Logs
Job Activity and System Events Logs

# Chapter 23: JAL Examples

The default installation of 24x7 Scheduler provides sample job database that includes examples of 24x7 script type jobs that you can use while you are learning Job Automation Language (JAL). This sample job database is distributed as part of the standard 24x7 installation package.

To see the JAL script for a job:
1 Double-click the job icon in the Job Explorer. The Job Wizard dialog box will appear.
2 Click the **Next** button, then click the **Edit** button. The JAL Editor window opens with the script in it.
3 After reviewing the script, close the Editor, then click the **Next** button again to see the actual job schedule. This is an important part of any scripting job.

Look at the following job examples:

| | |
|---|---|
| 1. | Running job between 9:00 AM and 5:00 PM every workday. |
| 2. | Scheduling job to not to run on particular days (job exception days). |
| 3. | Converting dates to filenames. |
| 4. | Converting comma-separated data file to tab-separated file. |
| 5. | Scanning program log file for errors. |
| 6. | Processing files using file mask. |
| 7. | Collecting Web server performance statistics. |
| 8. | Monitoring database server free space. |
| 9. | Monitoring file server free space. |
| 10. | Establishing dial-up connection, avoiding line drops. |
| 11. | Loading data into database. |
| 12. | Starting NT service, executing job, stopping service. |
| 13. | Paging/notifying system administrators. |
| 14. | Restoring network connection. |
| 15. | Unattended server reboot on demand. |
| 16. | Using FTP commands. |
| 17. | Redirecting program output to a file. |
| 18. | Verifying overnight data feed. |
| 19. | Using Windows messages. |
| 20. | Watching for file changes. |

## Running job between 9:00 AM and 5:00 PM every workday

```
Dim today, date
Dim time_now, time
Dim is_work_time, boolean
Dim is_holiday, boolean


CHECK_WEEKDAY:
Today( today )
isWeekday( today, is_work_time )
If( is_work_time, CHECK_HOLIDAY, DONE )

CHECK_HOLIDAY:
isHoliday( today, is_holiday )
If( is_holiday, DONE, CHECK_TIME )

CHECK_TIME:
Now( time_now )
isTimeBetween( time_now, 9:00, 17:00, is_work_time )
If( is_work_time, SOMETHING, DONE )

DO_SOMETHING:
// Perform the actual job here
```

```
// ...

DONE:
```

**See also:**
Other JAL script examples

## Redirecting program output to a file

```
Dim process_id, number
Dim command, string
Dim output, string
Dim position, number
Dim err_found, boolean

// For this example dynamically create batch file that includes just one command
FileSave( "redir.bat" , "ver" )

// Run the batch and terminate it after 30 seconds, in case
// if it does not stop. Batch output is redirected from the
// console to the out.txt file
RunAndWait  ("redir.bat >> out.txt" , "", 30, process_id)

// Read the output file for error checking
FileReadAll ("out.txt" , output)
Lower (output, output)
Pos (output, "error" , 1, position)
IsGreater (position, 0, err_found)
If (err_found, NOTIFY, END)

NOTIFY:
// Send e-mail message
MailSend("Exchange Settings" , "syspassword" , "admin@mycompany.com" , "Error occurred
while running DOSTEST.BAT" )

END:
// Done
```

**See also:**
Other JAL script examples

## Converting comma-separated data file to tab-separated file

```
Dim pointer, number
Dim buffer, string
Dim found, boolean

// Read comma separated file
FileReadAll( "sometext.txt" , buffer )

// Replace commas with tabs
Pos( buffer, ",", 1, pointer)
isGreater( pointer, 0, found)

LoopWhile( found, ENDLOOP )
   Replace( buffer, pointer, 1, "    ", buffer )
   Pos( buffer, ",", pointer, pointer)
```

```
   isGreater( pointer, 0, found)
ENDLOOP:

// Write tab separated file
FileSave( "tabtext.txt" , buffer )
```

**See also:**
Other JAL script examples

## Using FTP commands

```
Dim process_id, number
Dim found, boolean

// Watch for file on remote FTP site.
FTPFileExists( "ftp.microsoft.com" , "", "", "disclaimer.txt" , found )

// If the file found,
// continue processing, otherwise exit and wait for the next cycle
if (found, DOWNLOAD, END )

DOWNLOAD:
// Download the file from Microsoft FTP site
FTPGetFile( "ftp.microsoft.com" , "", "", "disclaimer.txt" , "c:\\temp\disclaimer.txt"
)
// Do something with the downloaded file, for example you can display
// it in the Notepad
Run( "notepad c:\\temp\disclaimer.txt" , "", process_id )

// Delete the file - in a real-world you most likely will do this
// FTPDeleteFile( "ftp.microsoft.com" , "", "", "disclaimer.txt" )

END:
// Done
```

**See also:**
Other JAL script examples

## Scheduling a job not to run on particular days (exception days)

One of the solutions is to create a text file of exception days so that every time the job runs it compares the current date with days in the exception file. For example, create the “ex_dates.txt" file using the text that follows:

03/01/98
03/08/98
04/01/98
04/08/98

Now create the new job with 24x7 script type and schedule this job to run daily. The following script can be used to check dates:

```
Dim buffer, string
Dim found, boolean
Dim today, date
Dim st_today, string
Dim pointer, number
```

```
// Read comma separated file
FileReadAll( "ex_dates.txt" , buffer )

// Get today’s date and convert it to a string
// Replace commas with tabs
Today( today )
Format( today, "mm/dd/yy" , st_today)

Pos( buffer, st_today, 1, pointer)
isGreater( pointer, 0, found)

if( found, END, DO_SOMETHING )

DO_SOMETHING:
// Do something here
// ...

END:
```

**See also:**

Other JAL script examples

## Converting dates to file names

Sometimes you may need to calculate file name as a function of date. You can use the following example as a template:

```
Dim today, date
Dim yesterday, date
Dim file_name, string

// Get today's date
Today( today )
// Calculate yesterday's date
DateAdd( today, -1, yesterday )
// Convert to string in mmddyyyy format
Format( yesterday, "mmddyyyy" , file_name )
// Append file extension
Concat( file_name, ".dat" , file_name )

// Do something with this file,
// for example FTP, Copy, Load into database, etc.
// ...
```

In some cases, you may use available macro-parameters to simplify date calculations. For example, the following script produces the same end-result as the script above:

```
Dim file_name, string

// Build file name
Concat( @DP" mmddyyyy" , ".dat" , file_name )

// Do something with this file,
// for example FTP, Copy, Load into database, etc.
// ...
```

**See also:**
Other JAL script examples

## Scanning program log file for errors

```
Dim buffer, string
Dim position, number
Dim err_found, boolean

// Load the log file into memory
FileReadAll ("program.log" , buffer)
// Convert to lower case to find all sorts of errors
Lower (buffer, buffer)
// Search for word "error"
Pos (buffer, "error" , 1, position)
// If error found, notify system administrator
IsGreater (position, 0, err_found)
If (err_found, NOTIFY, END)

NOTIFY:
// Send e-mail message
MailSend( "Exchange Settings" , "syspassword" , "admin@mycompany.com" , "Error
occurred while running PROGRAM.EXE. See program log file for details." )

END:
// Done
```

**See also:**
Other JAL script examples

## Processing files using file mask

```
Dim file_name, string
Dim found, boolean
Dim attr, number
Dim read_only, boolean
Dim hidden, boolean
Dim system, boolean
Dim directory, boolean
Dim archived, boolean
Dim message, string

// Start file search
FileFindFirst( "*.txt" , file_name, found )

LoopWhile( found, ENDLOOP )
   // File found, do something with the file here
   // ...
   // For example we can get file attributes then display them
   FileGetAttr( file_name, attr )
   BitwiseGetBit( attr, 1, read_only )
   BitwiseGetBit( attr, 2, hidden )
   BitwiseGetBit( attr, 3, system )
   BitwiseGetBit( attr, 5, directory )
   BitwiseGetBit( attr, 6, archived )

   // skip directories
   IfThen( directory, FIND_NEXT )
```

```
   // Show nice message
   Char( 13, new_line )
   Concat( "File: ", file_name, message )
   Concat( message, "\n\tRead Only: ", message )
   Concat( message, read_only, message )
   Concat( message, "\n\tHidden: ", message )
   Concat( message, hidden, message )
   Concat( message, "\n\tSystem: ", message )
   Concat( message, system, message )
   Concat( message, "\n\tDirectory: ", message )
   Concat( message, directory, message )
   Concat( message, "\n\tArchived: ", message )
   Concat( message, archived, message )
   MessageBox( message )

   FIND_NEXT:
   // Find next file
   FileFindNext( file_name, found )
ENDLOOP:
```

**See also:**

Other JAL script examples

## Collecting Web server performance statistics

```
// This script allows checking Web server response time.
// The number is saved in a file, which can be helpful
// in analyzing Web server performance such as estimating
// average response time and building busy rate graph as
// a dependency of time of day.

Dim current_time, time
Dim current_date, date
Dim start_time, datetime
Dim end_time, datetime
Dim duration, number
Dim file_number, number
Dim line, string


// Get start time
Today( current_date )
Now( current_time )
DateTime( current_date, current_time, start_time )

// Download index page from a Web server
// For example use popular Microsoft Web site
WebGetPageHTML( "http://home.microsoft.com/", "c:\\temp\msweb.tmp" )

// Get end time
Today( current_date )
Now( current_time )
DateTime( current_date, current_time, end_time )

// Calculate response time
DateTimeDiff( start_time, end_time, duration )

// Build text line
Concat( "Start: ", start_time, line )
Concat( line, "  End: ", line )
Concat( line, end_time, line )
Concat( line, "  Duration: ", line )
Concat( line, duration, line )
```

```
// Append built line to the statistics file
FileOpen( "webstat.txt" , "LineMode" , "Write" , TRUE, file_number )
FileWrite( file_number, line )
FileClose( file_number )

// Uncomment next line to open statistics file in Notepad
// Run( "notepad webstat.txt" , "", file_number )
```

**See also:**

Other JAL script examples

## Monitoring file-server free space

```
// Before trying this script, correct the
// network drive letter specified below, if necessary

Dim drive, string
Dim free_space, number
Dim low_space, boolean

Set( drive, "K" )
// Get free space
// Windows95 OSR-1 users use DiskGetFreeSpace instead of DiskGetFreeSpaceEx
DiskGetFreeSpaceEx( drive, free_space )
// Check threshold - 50 Mbytes = 50 * 2^20
isLess( free_space, 52428800, low_space )

if( low_space, PROBLEM, END )
PROBLEM:
// Page network administrator
MailSend( "Exchange Settings" , "system" , "12345678@pagenet.com" , "Low of space" ,
"Free space on drive K below 50 Mbytes" )

END:
```

**See also:**

Other JAL script examples

## Using Windows messages

```
// This example was developed to demonstrate how to send
// Windows messages from a script.
Dim process_id, number
Dim window_handle, number
Dim edit_handle, number

// Use Notepad for demonstration
// Run Notepad
Run( "notepad" , "", process_id )
// Wait 2 seconds for the Notepad to open
Wait( 2 )
// Get handle of the Notepad window
ProcessGetWindow( process_id, window_handle )
// Get handle of the edit box
WindowGetChild( window_handle, edit_handle )
// Send EM_SETREADONLY message to the edit box
// to make it read only
WindowSendMessage( edit_handle, 207, 1, 0 )
```

```
// Open schedule log file in the Notepad
WindowActivate( window_handle )
SendKeys( "{ALT}FOschedule.log{ENTER}" )
```

**See also:**
Other JAL script examples

## Unattended server reboot on demand

```
// In this example, we assume that the 24x7 Scheduler is running on the server
// and that this job is setup as an "e-mail watch." On receiving the specified // e-
mail message, 24x7 Scheduler will reboot the computer. This script has
// just one line:

Reboot

// That's all, folks! Now imagine how easy it is to reboot your database and
// Web servers without leaving your home. You can also setup this job to run
// every day so that it will restart the server every morning at 7:00 AM before
// people come to the office.
```

**See also:**
Other JAL script examples

## Restoring network connection

```
// Use Windows Explorer to restore a disconnected network drive before
// running a program that accesses data files on the disconnected drive.

Dim process_id, number
Dim window_handle, number
Dim running, boolean

Run( "explorer k:\data" , "", process_id )
// Wait 10 seconds to allow the Explorer to restore connection
Wait( 10 )

// First let’s try to close Explorer gracefully (this is always better than killing
the process.)
// However, using the ProcessKill statement guarantees that the process termination. A
keystroke, on the other hand,
// is sent to an active window, which can change at anytime.
SendKeys( "{ALT}{F4}" )

// Wait 2 more seconds
Wait( 2 )
// Check whether Explorer is still running or not
ProcessGetWindow( process_id, window_handle )
isGreater( window_handle, 0, running)

if( running, KILL, DO_MAIN_JOB )
KILL:
ProcessKill( process_id )

DO_MAIN_JOB:
// Do the main job here
// Run ...
```

**See also:**
Other JAL script examples

## Watching for file changes

```
// This is an example of a "file change watch" job. A regular "file watch" job checks
// file presence only. In some situations, you may need to run a job only when
// a change occurs.

// This script uses the "change.txt" file to store "watch file" attributes between
runs.

Dim buffer, string
Dim file_time, string
Dim file_date, string
Dim date_time, datetime
Dim buffer2, string
Dim no_change, boolean

// Read parameter file into buffer
FileReadAll( "change.txt" , buffer )

// Get watch file date/time
FileTime( "k:\data\account.dat" , file_time )
FileDate( "k:\data\account.dat" , file_date )
DateTime( file_date, file_time, date_time )
// Convert to string
Format( date_time, "mm/dd/yyyy hh:mm:ss" , buffer2 )
// Compare file times
isEqual( buffer, buffer2, no_change )

// If the file date/time is the same, then nothing to do, otherwise
// update the parameter file and perform the main job.
If( no_change, END, UPDATE )
END:
Exit

UPDATE:
FileSave( "change.txt" , buffer2 )
// do main job here
// ...
```

**See also:**
Other JAL script examples

## Paging/notifying system administrators

The 24x7 Scheduler allows you to setup a job that can send a page or an e-mail message to one recipient at a time. Sometimes, if a job fails, it is necessary to notify more than one person. You can use JAL to add this function to virtually any job. One way to do this is to setup the “file type" notification action for the main “program" or “database" type job.. In the event of a failure, a semaphore file will be created (for example: job.err). Another job of 24x7 script type instantly checks presence of the specified semaphore file. As soon as the file is detected, the job deletes the file and sends pages or e-mail messages to the specified recipients. Note that the second job will need to be scheduled as a “file watch". The following example shows how to send multiple messages by using JAL e-mail statements.

```
// Note: The list of recipients can be hard-coded to simplify scripting.
// This list of recipients can be read from file. This was done in order to make the
script reusable.
// Thus recipients can be changed without changing the job definition.

Dim file_number, number
Dim recipient, string
Dim end_of_file, boolean

// Open recipients file for reading line by line
FileOpen( "mail.lst" , "LineMode" , "Read" , "", file_number )

// Read the file line by line until the end of file.
// Each recipient must be on a separate line.
EOF( file_number, end_of_file )
LoopUntil end_of_file, ENDLOOP
   FileRead( file_number, recipient )
   // Send e-mail
      MailSend( "Exchange Settings" , "syspassword" , recipient, "any subject" , "any
message" )

      EOF( file_number, end_of_file )
ENDLOOP:

// Done
FileClose( file_number )
```

**Notes:**

- There are many companies that provide reliable “e-mail to pager" services. For example: CompuServe, SkyTel, PageNet, MobileComm, and PageMart. There is no major difference between sending e-mail messages and sending messages to a pager. The only real difference is that when you send a message to a pager you must use a special addressing format. For example, the address format for the CompuServe is: CompuServe_ID@mobile.compuserve.com. If the account number (CompuServe ID) is 79999,9999, the recipient could receive wireless e-mail by using the address 79999.9999@mobile.compuserve.com. Note that a period must be used in place of a comma.
- In the case of an alphanumeric message, the number of characters received by the recipients depends on how they setup their pager preferences. Keeping your message short and concise helps to ensure that the recipient gets the most out of your message.

**See also:**

Other JAL script examples

# Monitoring database free space

This example shows how easily you can perform various database operations in 24x7 Scheduler. To simplify the script we saved the SQL part in a separate file called free_space.sql. The script will load this file and dynamically execute the loaded SQL. This SQL was designed for Oracle 7 databases. You may need to customize it for your database.

```
// Load and execute SQL script from "free_space.sql" file
// If the result is positive (at least one segment found),
// notify the database administrator about potential problems.

Dim SQL, string
Dim rows, number
Dim problem, boolean
Dim message, string
```

```
// Load file
FileReadAll( "free_space.sql" , SQL )
// Connect to database and retrieve "bad" segments,
// see "free_space.sql" file for details
DatabaseConnect( "Sales DB" )
DatabaseRetrieve( SQL, rows )
DatabaseDisconnect

isGreater( rows, 0, problem )

// If problem detected, notify DBA
If( problem, NOTIFY, END )

NOTIFY:
// Save retrieved data in the temporary file
DatabaseSave( "c:\\temp\message.tmp" , "TXT" , rows )
// Read temp. file contents and then e-mail to DBA
FileReadAll( "c:\\temp\message.tmp" , message )
MailSend( "Exchange Settings" , "pwrd" , "ora_dba@my_company.com" , "Database Problem"
, message )
// Delete temp. file
FileDelete( "c:\\temp\message.tmp" )

END:
```

**FREE_SPACE.SQL**

```
SELECT 'Owner: ' || seg.owner || chr(13) || chr(10) ||
   'Name: ' || seg.segment_name || chr(13) || chr(10) ||
   'Type: ' || seg.segment_type|| chr(13) || chr(10) ||
   'Tablespace: ' || seg.tablespace_name || chr(13) || chr(10) ||
   'Next extent size: ' || to_char(seg.max_extents, '999,999,999') || chr(13) ||
chr(10) ||
   'Problem: Max extent reached'
FROM sys.dba_segments seg
WHERE seg.extents = seg.max_extents

UNION ALL

SELECT 'Owner: ' || seg.owner || chr(13) || chr(10) ||
   'Name: ' || seg.segment_name || chr(13) || chr(10) ||
   'Type: ' || seg.segment_type|| chr(13) || chr(10) ||
   'Tablespace: ' || seg.tablespace_name || chr(13) || chr(10) ||
   'Next extent size: ' || to_char(t.next_extent, '999,999,999') || chr(13) || chr(10)
||
   'Problem: Not enough space for next extent'
FROM sys.dba_segments seg,
   sys.dba_tables t
WHERE seg.segment_type = 'TABLE'
   AND seg.segment_name = t.table_name
   AND seg.owner = t.owner
   AND NOT EXISTS (SELECT tablespace_name
   FROM dba_free_space free
   WHERE free.tablespace_name = t.tablespace_name
   AND free.bytes >= t.next_extent
   )
UNION ALL

SELECT 'Owner: ' || seg.owner || chr(13) || chr(10) ||
   'Name: ' || seg.segment_name || chr(13) || chr(10) ||
   'Type: ' || seg.segment_type|| chr(13) || chr(10) ||
   'Tablespace: ' || seg.tablespace_name || chr(13) || chr(10) ||
```

```
    'Next extent size: ' || to_char(i.next_extent, '999,999,999') || chr(13) || chr(10)
||
    'Problem: Not enough space for next extent'
FROM sys.dba_segments seg,
    sys.dba_indexes i
WHERE seg.segment_type = 'INDEX'
    AND seg.segment_name = i.index_name
    AND seg.owner = i.owner
    AND NOT EXISTS (SELECT tablespace_name
    FROM dba_free_space free
    WHERE free.tablespace_name = i.tablespace_name
    AND free.bytes >= i.next_extent
    )
UNION ALL

SELECT 'Owner: ' || seg.owner || chr(13) || chr(10) ||
    'Name: ' || seg.segment_name || chr(13) || chr(10) ||
    'Type: ' || seg.segment_type|| chr(13) || chr(10) ||
    'Tablespace: ' || seg.tablespace_name || chr(13) || chr(10) ||
    'Next extent size: ' || to_char(c.next_extent, '999,999,999') || chr(13) || chr(10)
||
    'Problem: Not enough space for next extent'
FROM sys.dba_segments seg,
    sys.dba_clusters c
WHERE seg.segment_type = 'CLUSTER'
    AND seg.segment_name = c.cluster_name
    AND seg.owner = c.owner
    AND NOT EXISTS (SELECT tablespace_name
    FROM dba_free_space free
    WHERE free.tablespace_name = c.tablespace_name
    AND free.bytes >= c.next_extent
    )
UNION ALL

SELECT 'Owner: ' || seg.owner || chr(13) || chr(10) ||
    'Name: ' || seg.segment_name || chr(13) || chr(10) ||
    'Type: ' || seg.segment_type|| chr(13) || chr(10) ||
    'Tablespace: ' || seg.tablespace_name || chr(13) || chr(10) ||
    'Next extent size: ' || to_char(r.next_extent, '999,999,999') || chr(13) || chr(10)
||
    'Problem: Not enough space for next extent'
FROM sys.dba_segments seg,
    sys.dba_rollback_segs r
WHERE seg.segment_type = 'ROLLBACK'
    AND seg.segment_name = r.segment_name
    AND seg.owner = r.owner
    AND NOT EXISTS (SELECT tablespace_name
    FROM dba_free_space free
    WHERE free.tablespace_name = r.tablespace_name
    AND free.bytes >= r.next_extent
    )
```

**See also:**

Other JAL script examples

## Starting NT service, executing job, stopping NT service

Some Windows NT (NT/2000/XP/2003/VISTA/2008/7) applications are designed as Windows NT services. In some cases it is not necessary to keep such application running all the time, as that application will take some computer and Operation System resources leaving less resources for other applications. This example shows how to manage the Windows NT service from JAL.

```
// This script starts Personal Oracle 8.0, runs a program
// that loads information into the database, then the script
// shutdowns the database.

// Start Oracle database. Note that service OracleStartORCL automatically
// starts another service OracleService.
ServiceStart "OracleStartORCL"
ServiceStart "OracleTNSListener80"
ServiceStart "OracleClientCache80"

// Do main job here
// ... Run ...

// Shutdown database
ServiceStop "OracleClientCache80"
ServiceStop "OracleTNSListener80"
ServiceStop "OracleStartORCL"
ServiceStop "OracleService"
```

**See also:**

Other JAL script examples

## Loading data into database

```
// This script imports data into a temporary table called TEMP_SALES_ORDER,
// then populates the main SALES_ORDER table from the temporary table.
Dim imported_rows, number
Dim old_rows, number
Dim new_rows, number
Dim message, string

// Connect to database
DatabaseConnect( "Sales DB (Oracle 7.2)" )
// Prepare temp.table
DatabaseExecute( "TRUNCATE TABLE scott.temp_sales_order" , old_rows )
// Populate temp. table
DatabaseImport( "scott.temp_sales_order" , "order.txt" , imported_rows )
// Delete from main table matching order to avoid duplicate key problem
DatabaseExecute( "DELETE FROM scott.sales_order WHERE id IN (SELECT id FROM
scott.temp_sales_order)", old_rows )
// Populate main table
DatabaseExecute( "INSERT INTO scott.sales_order SELECT * FROM scott.temp_sales_order"
, imported_rows )
// Disconnect from database and commit transaction
DatabaseDisconnect


// Notify job owner about results
Subtract( imported_rows, old_rows, new_rows )
Concat( "Updated orders: ", old_rows, message)
Concat( message, ", New orders: ", message)
Concat( message, new_rows, message)

MailSend( "Exchange Settings" , "pswrd" , "sctott@mycompany.com" , "Sales Order
Update" , message )
```

**See also:**

Other JAL script examples

# Establishing dial-up connection, avoiding line drops

```
// This script demonstrates how to initiate a dial-up connection from JAL
// script. To use a different connection name, replace CompuServe with
// the name of your dial-up connection. RnaDial and your connection name
// are case-sensitive, so match the text exactly. Make sure to setup your
// dial-up connection so that it remembers the password and does not have to prompt
for it.

Dim process_id, number
Dim window_handle, number

// Connect to CompuServe
Run( "rundll rnaui.dll,RnaDial CompuServe" , "", process_id )
// Wait 60 seconds to allow dial-up networking to initiate the connection
Wait( 60 )
// do something here
// ...
// done

// To disconnect from CompuServe, close the dial-up window.
// To leave the connection online, comment the following two lines
WindowFind( "Connect To%", window_handle )
WindowClose( window_handle )

// If you do not want to close the connection, but keep this
// connection online, even with long periods of inactivity, you can use
// the ping command. setup a new job that will run All Day every minute.
// Select the "program" type for this job and specify the ping command in the
// program name field. This will help avoid line drops.
// For example: ping www.ibm.com
```

**See also:**

Other JAL script examples

# Verifying overnight data feed

```
// Check data files that arrived earlier.
Dim file_date, date
Dim file_time, time
Dim today, date
Dim now, time
Dim not_found, boolean
Dim file_OK, boolean
Dim date_OK, boolean
Dim time_OK, boolean
Dim file_name, string

Today( today )
Now( now )

CHECK_ACCOUNTS:
// Get date/time for Accounts
Set( file_name, "j:\data\account.dat" )
NotFileExists( file_name, not_found )
IfThen( not_found, BAD_FILE )
FileDate( file_name, file_date )
FileTime( file_name, file_time )

// Check if it is file_OK
IsEqual( file_date, today, date_OK )
```

```
IsTimeBetween( file_time, 00:00:00, now, time_OK )
And( date_OK, time_OK, file_OK )
if( file_OK, CHECK_HOLDINGS, BAD_FILE )


CHECK_HOLDINGS:
// Get date/time for Holdings
Set( file_name, "j:\data\holding.dat" )
NotFileExists( file_name, not_found )
IfThen( not_found, BAD_FILE )
FileDate( file_name, file_date )
FileTime( file_name, file_time )

// Check if it is file_OK
IsEqual( file_date, today, date_OK )
IsTimeBetween( file_time, 00:00:00, now, time_OK )
And( date_OK, time_OK, file_OK )
if( file_OK, CHECK_SECURITIES, BAD_FILE )


CHECK_SECURITIES:
// Get date/time for securities
Set( file_name, "j:\data\secirity.dat" )
NotFileExists( file_name, not_found )
IfThen( not_found, BAD_FILE )
FileDate( file_name, file_date )
FileTime( file_name, file_time )

// Check if it is file_OK
IsEqual( file_date, today, date_OK )
IsTimeBetween( file_time, 00:00:00, now, time_OK )
And( date_OK, time_OK, file_OK )
if( file_OK, DONE, BAD_FILE )

BAD_FILE:
// Inform operation personal about bad file
MailSend( "Exchange Settings" , "pswrd" , "david@mycompany.com" , file_name,
"Overnight feed failed. File specified in the subject was not updated!" )
MailSend( "Exchange Settings" , "pswrd" , "joe@mycompany.com" , file_name, "Overnight
feed failed. File specified in the subject was not updated!" )

DONE:
```

# Chapter 24: Technical Support

Your questions, comments, and suggestions are welcome.

For technical support, e-mail to support@softtreetech.com or use the on-line support form at http://www.SoftTreeTech.com/Support.htm.

When reporting problems, please provide as much information as possible about your problem. Be sure to include the following information:

1 Is the problem reproducible? If so, how?
2 What version of Windows are you running? For example, Windows 95, Windows NT 4.0, etc.
3 What version of the 24x7 Scheduler are you running?
4 If a dialog box with an error message was displayed, please include the full text of the dialog box, including the text in the title bar.
5 If the problem involves an external program, provide as much information as possible about this program.
6 Make sure you include the serial number for your copy of 24x7 Scheduler. Use the **Help/About** menu to look up the correct numbers. Registered users have priority support.

For registration information, purchasing or other sales information, please contact our sales department: sales@softtreetech.com.

For general information, software updates, the latest information on known problems and answers to frequently asked questions, visit the 24x7 Scheduler home page on the Web: http://www.SoftTreeTech.com/24x7/.

We are happy to help in any way we can, but if you are having problems, please check the troubleshooting section first to see if your question is answered there.

# Chapter 25: Frequently Asked Questions (FAQ)

**Q: I created a JAL script job, which I want to run one of my programs (or copy file, or rename file, or perform other file operation…). I am sure the file referenced in the script exists but when I run my job it fails with an error "The system cannot find the file specified". What's wrong?**

**A:** In the file names referenced in your JAL script you are not using double-backslashes instead of single-backslashes. As a result of that some special literals in the script, such as \t, \n, \r and some other are replaced in run-time with the **special ASCII characters**. To fix this problem, edit your script and replace every backslash character with double-backslash. For example, specify file "c:\temp\somefile.txt" as "c:\\temp\\somefile.txt". For more information please see "Special ASCII characters" topic.

**Q: I have setup a job that should run unattendant. This job should send an email messages in case of an error. When the job fails, it creates an email message, but the message is not sent and the "New message" window remains on the screen. How do I fix it?**

**A:** You are using MAPI email interface. Different email programs utilize different MAPI message properties. That's why 24x7 Scheduler provides configuration options that allow you to tune it for your email program. The email will not be send if you have incompatible properties selected. To fix this problem, select Tools/Options menu. The Options dialog will appear. Change email send method to "send by name". Either "send by name" or "send by address" option should be compatible with your email program.

**Q: After I have created a new job and then clicked the Save button on the toolbar, the job became disabled. What's wrong?**

**A:** You did not specify at least one mandatory property for that job. See job log for details on invalid properties. Most likely, you forgot to specify the job start time. Double-click the job's icon, then correct the job schedule. Release the Disable button on the toolbar. Save changes.

**Q: How can I setup my program to run every workday from 9:00 AM to 5:00 PM?**

**A:** Create a new job of the 24x7 script type. Schedule this job to run "All Day". Put in the script some logic to check the desired time condition. If the condition is satisfied, use the Run statement to start the desired program.

**Q: How can I setup a "Fix It" job?**

**A:** Activate the "Error" Notify Event for the primary job. setup either "File" or "E-mail" Notify Action. setup your "Fix It" job as a "file watch" or "E-mail watch" correspondingly. Specify "watch" condition matching parameters of the Notify Action that the 24x7 Scheduler will perform in case the primary job fails.

**Q: I am using just one computer and I want to run some of my "long running" database and 24x7 script jobs in the background. How do I do that?**

**A:** Turn on the "asynchronous job" property.

**Q: Is there a way to save the result returned by a database job?**

**A:** Do not use a database type job for this sort of task. Instead, create a 24x7 script type job. Use DatabaseConnect, DatabaseRetrieve, DatabaseSave, and DatabaseDisconnect statements to connect to the database, retrieve data, save the data in an external file, and then end the database connection.

**Q: After each job execution, 24x7 Scheduler leaks some memory. Why is that?**

**A:** For performance reasons 24x7 Scheduler keeps a copy of the job execution log loaded into computer memory. While executing a job, the 24x7 Scheduler may create new entries in the job execution log. These new entries take some additional memory chunk, until the log reaches the maximum number of allowed entries specified in the system preferences. After that, the most-recent entry pushes the oldest entry out from the log. The memory "leakage" process will then stop. In order to minimize the amount of memory required for the job log you may want to do the following:

a. Decrease the maximum number of allowed log entries.
b. Disable logging for jobs that run often and it is not important to you whether such jobs were successful or not.

**Q: I have difficulties running old DOS applications asynchronously. As soon as the screen saver takes over, my DOS job stops until I click on the DOS window!**

**A:** Screen Savers prevent DOS Applications from running in the background. One way to avoid this problem is to use a PIF file to run the application. In the PIF file editor check the "Allow background processing" and “run in a window”. This allows the application to act more like a window and continue its process.

**Q: I have installed 24x7 Scheduler on my new server. Can I copy all existing settings from the old installation?**

**A:** Yes! See **Installation and Uninstallation** topic for details.

**Q: How does 24x7 Scheduler know if something ran successfully?**

**A:** When a program job is setup to run synchronously, 24x7 Scheduler checks the "Exit Code" for that program after the job completes. Exit Code values are used by many programs to indicate if they ran successfully. There is no way for the 24x7 Scheduler to find out an Exit Code for the program that you setup to run asynchronously. The 24x7 Scheduler simply manages to start such program and does not know when and how the program completes. Some programs create an application log file during the programs run. You may also try to run such programs using 24x7 script, then search the application log file for common error messages. For a database job, 24x7 Scheduler always checks the "SQL Code" that the database returns after processing. This code reports whether the performed database operation ran successfully or not.

**Q: I would like to run several programs sequentially in one job. How do I set it up?**

**A:** There are two ways to do this. One way is to create a .BAT file that runs all the desired application one by one, then schedule this .BAT file as a single job. Another way is to create a 24x7 script type job, then use the JobRunAndWait statement to implement the desired batch logic.

**Q: I have scheduled a job that kicks off the DOS BAT file. After the .BAT file completes, the DOS window remains open. How can I avoid this?**

**A:** You have two options:
a. Run this BAT file manually. Select **Properties** from the window's control menu. Check the **Close on Exit** option.
b. Specify the command line for this program as *CMD.EXE /c C:\MYPATH\MYFILE.BAT*. The /c tells the system CMD.EXE command to exit after processing the MYFILE.BAT file. For more help on the CMD command, type CMD /? at the command prompt. On Windows 95/98/Me use the command *COMMAND* instead of *CMD.*

**Q: How do I know which job and when the 24x7 Scheduler will run next?**

**A:** You can rest the mouse pointer over the third section of the Status Bar to view this information. Double-click on this section to locate and highlight the very first pending job. For more details see **Job Explorer** topic. You can also use the **Job Monitor** to get the list of all pending jobs in the 24-hour timeframe.

**Q: How can I configure a program to run after the data it depends on is available?**

**A:** Setup this job schedule as a "file watch". Specify one or more data files in the **semaphore files** property. Set a reasonable polling interval at which the 24x7 Scheduler will check for the presence of specified files.

**Q: I have two 24x7 Schedulers installed in two offices. One of them is watching for a file. As soon as the file arrives, I would like it to be e-mailed to another office where the second scheduler will upload this file to a database. How can I do that?**

**A:** Setup a "file watch" job using "Run 24x7 Script". job type Code the SendMailWithAttachment statement to transmit the file to another office. In the destination office, setup an "e-mail watch" job of "Run 24x7 Script" type. Code the "upload" logic using available file and database statements.

# Index

## D

## E



## F

## G



## H



## I



## J

## K



## L



## M

## N



## O



## P

## Q



## R



## S

## T

## U



## V



## W



## Y



## Z