

Version 6.x

Language

[English](#)

Contents

- [Introduction](#)
- [Purchase Information](#)
- [Getting Started](#)
- [Basics](#)
- [Tasks](#)
- [Variables](#)
- [Tutorials](#)
- [NT Service Module](#)
- [Command Line Module](#)
- [User Case Studies](#)
- [Performance and Reliability Testing](#)

Introduction



This software is designed to automatically execute tasks. The software can be run on windows, macintosh, linux, and on other operating systems.

Tasks can be scheduled on a daily, weekly, monthly, or hourly basis. Tasks can also be scheduled by the minute, or second. Tasks include local and internet applications, file download and monitoring.

This software is available on a 30 day free trial basis. You can test it during this period. After 30 days, the software expires, and you will need to register the software. If you do not want to use the software, you can uninstall it from your computer. The uninstall procedure will depend on the operating system. Typically, an uninstall program is also loaded onto your computer. Please see your help file for uninstall instructions.

If you need to use this software for more than 30 days, please register it. You can purchase online by credit card, check, money order, or purchase order. Multiple copy discounts, and site licenses are available. See the Hitek Software web page, at "<http://www.hiteksoftware.com>" for details.

Purchase Information

When you purchase this software, you will receive registration instructions via email. Follow the registration instructions very carefully, to receive the registration key. The registration key is required to unlock the software.

Please visit our web site at '<http://www.hiteksoftware.com>', for latest purchasing details and pricing information. You can purchase, using one of the following methods:

1. Credit Card online. The registration instructions are automatically sent to you via Email.
2. Phone in or fax your order. Please see our web site for details
3. Send a check or money order. Please see our web site for details.
4. Purchase Order. Processing fees will apply for purchase orders.

Multiple copy discounts and site licenses are available. See the Hitek Software web site at '<http://www.hiteksoftware.com>' for details.

Getting Started

Creating a new task

From the front panel, select the 'Tasks' menu.

From the pull down menu, select the 'Administrative' menu. Then select the 'Archive Logs' menu item.

This task archives the programs output and activity logs.

Enter a Title for this 'Archive Logs' task. Select a local directory to archive the logs.

Click on the 'Save' button. The task will appear in the 'All Tasks' table.

Running a task

Select the task in the 'All Tasks' table, and click the 'Run' button. 

The task will be run by the Scheduler Engine.

Click on the Logs menu of the front panel. Then click on the 'Output Log' menu item. You will see the output from the Archive Logs task.

Scheduling a task

Select the task in the 'All Tasks' table, and hit the 'Schedule' button. 


The scheduler box will pop up. Set the schedule type as 'Minute'.

Hit 'Save' to save this schedule. The task will appear in the 'Scheduled Tasks' table.


Wait for a minute, and the 'Archive Logs' task will run again. This task will run every minute. In the output log viewer, click the Refresh Button, to view the latest Run output.

You can stop the schedule, by disabling the 'Run' check box, in the first column of the 'Scheduled Tasks' table.

Editing a task

To edit your task, select the task in the 'All Tasks' table, and click the 'Edit' button.  You can change your task parameters, and hit 'Save' again. The original task data will be updated.


Editing a task schedule

To edit your task, select the task in the 'Scheduled Tasks' table, and click the 'Edit' button.  You can change your schedule parameters, and click the 'Save' button. The original schedule data will be updated.

Deleting a task

To delete the task, select the task in the 'All Tasks' table, and click the 'Delete' button. 

Deleting a task schedule

To delete the schedule for a task, select the task in the 'Scheduled Tasks' table, and click the 'Delete' button. 

Basics

- [Getting Started](#)
- [Using Help](#)
- [Front Panel](#)
- [Task Table](#)
- [Schedule Table](#)
- [Scheduling Tasks](#)
- [Exit Codes](#)
- [Log Files](#)
- [Append to Filename](#)
- [Filename Filter](#)
- [Scheduler Settings](#)
- [Unix Service](#)

Tasks



Below is a list of tasks, which are supported by the 6.x family of Hitek Software products. Please note that all tasks listed below may not be available in your program. For example, the Command task is available in Automize 6.x, but not in AbleFtp 6.x.

<u>Archive Logs</u>	<u>Auto Backup</u>	<u>Bulk Email</u>
<u>Command</u>	<u>Chains</u>	<u>Copy Files</u>
<u>Check Email</u>	<u>Database SQL</u>	<u>Directory Change</u>
<u>DeleteFiles</u>	<u>Database Test</u>	<u>Directory Monitor</u>
<u>Email</u>	<u>Email Logs</u>	<u>File Monitor</u>
<u>Ftp</u>	<u>Ftp Commands</u>	<u>FtpMonitor</u>
<u>File Variable</u>	<u>LogFile monitor</u>	<u>Maintenance</u>
<u>Ping</u>	<u>Print Files</u>	<u>Synchronize</u>
<u>TextSearch</u>	<u>Telnet</u>	<u>Telnet(Adv)</u>
<u>Unzip Archive</u>	<u>Url Monitor</u>	<u>Variable Monitor</u>
<u>Web Downloads</u>	<u>WinCommand</u>	<u>ZipDirectories</u>

Variables

This software supports variables using two techniques:

- 1) Many fields supporting variables: The value of these variables is resolved when the task runs. The fields, which support variables, are marked with **\$**. For example: to match files in a Filename field based on current month of year, you would enter the filename as: `$(DATE)::mm-yy%`. If the current month was June 2002, all files, which contain '06-02' in their name, would be selected.
- 2) [Variable Monitor](#) task: This task compares the value of the variable using a value, and a comparison criteria you define. If a match is found, you can run other tasks.

Types of variables

Task Variables: Task variable values are generated/updated when the tasks run. Example: Tasks exit codes are stored as the following variable: `TaskTitle::EXITCODE`

System Variables: System variables which are resolved when the current task runs, examples are DATE, USERNAME etc..

Java Variables: Java variables, which are available to the Java Virtual Machine. You can get the value of java variables from the 'Java Variables' menu item

User Variables: Users can add their own variables via batch files, scripts or programs. Users can link output from their own programs, to this program.

Using variables in the text fields

The variable must be formatted as follow: `$(VariableFormat)%`

For System variables: `$(VariableName::VariableOptions)%`

Variable_Options may only be required by some Variables.

examples: `$(DATE)::hh:mm;yyyy%` or `$(USERNAME)%`

For Task variables: `$(TaskTitle::VariableName)%`

examples: for TaskTitle = Test : `$(Test::ExitCode)%` or `$(Test::ErrorMsg)%`

For User and Java variables: `$(Variable)%`

examples: for java variable user.name, use : `$(user.name)%`

List of currently available variables

To view the list of currently available variables, click on the 'Variable Monitor' task dialog (via Tasks

menu / Monitors). The variables are listed in the Variable Name dropdown box. The variables are listed in the following order:

- 1) System variables
- 2) Task and user variables
- 3) Java variables

Tutorials

- [How To's](#)
- [Windows specific tutorials](#)

NT Service Module 6.x - Please read this entire file carefully

The NT service module is an optional add-on. It requires a separate download and purchase. Please visit our website to download the NT service module. This module can only be used with version 6.0 or later. The NT service module launches the Scheduler Engine, if it is shut down. When a user logs out, the scheduler engine will shut down. The NT service module will restart the engine after logout, and periodically verify that the engine is always running. This module will also launch the engine on startup, if the Service Startup Type is set to Automatic (default).

[Getting Started](#)

[Service Control Summary](#)

[Recommended settings in Service Control Manager](#)

Getting Started

1. Install the NT service using the Service menu, from the module front panel. The NT Service Module window does not need to be kept open. You will only use the module, to register the software, change settings, or remove the service.
2. Go to the Windows Service Control Manager, via the Control Panel. Set the logon properties for the NT Service, as shown in the Recommended Settings table below. Start the NT Service.

Notes:

1. For version 6.x+, the default scheduler engine Port = 1965. **DO NOT** change this value, unless you have also changed the engine port value, in the software also. **The port value in the NT service module, should match the port value in the user interface.**
2. The Period is the time interval in minutes, which the NT service module, will check that the engine is running. The default is 5 minutes. **When a user logs out, the scheduler engine will shut down. The Service will then re-launch the engine within the Period set (default = 5 minutes).**
3. If you change the Port value, or the Period value, from the NT Service module, the Service needs to be restarted, from the Service Control Manager.

Service Control Summary

Install Service	Use NT Service Module
Remove Service	Use NT Service Module
Start Service	Use Service Control Manager (from control panel)
Stop Service	Use Service Control Manager (from control panel)
Change Port or Period settings	Use NT Service Module
Register Software (unlock software)	Use NT Service Module

Recommended settings in Service Control Manager

Account	<p>User Account - You must enter your username/password and have administrative privileges.</p> <p>If you do not satisfy the above conditions, you will get the following error: Service did not startup or respond in a timely manner.</p>
Interact with desktop	NO (unchecked)
Startup Type	Automatic (default)

Command Line Module



[Introduction](#)

[Using the Command Line Module](#)

[Command Reference](#)

[Adding/Modifying Tasks](#)

[List of parameters for each task type](#)

[Adding/Modifying Schedules](#)

[Installing Software to a remote System](#)

[Using Ftp to set/view remote data](#)

[Managing remote scheduler\(s\)](#)

[Programmatically controlling the scheduler](#)

Case Studies

These are actual examples of how users are using our software. We obtain this information when users request technical support. You can learn valuable tips by going through these examples. Note that some of these examples only apply to Automize and not other programs (i.e. AbleFtp or JaSFtp etc..)

[Scheduling a task to run on the first Monday of the month](#)

[Monitoring a website/server - receive alert when server is down](#)

[Monitoring a website/server using reference - receive alert when server is down](#)

[Monitor a log file for alerts and receive immediate cell phone notification](#)

[Monitoring log files for errors / exit codes](#)

[Detect errors in telnet tasks using variable monitor](#)

[Save results of a telnet command to local file](#)

[Daemon to monitor directory for changes and run file processing program](#)

[File Monitor task to monitor directory for total accumulated files](#)

[Run task if file exists in a directory, else send an email](#)

[Dynamically download URLs that are specified in a text file](#)

[Delete files from Ftp server directory that are older than 30 days](#)

[When new files come into folder, send email and attach these files](#)

[Monitor email server: ensure it is processing incoming and outgoing emails](#)

[Backup a file and receive email notification of success/failure](#)

[Zip each file in a folder into its own archive](#)

[Download webpages and append them to master file](#)

Reliability and Performance Tests - version 6.x

General

Automize is now 100% java and uses Sun's Java 1.3 machine. If it runs on one system, it should **supposedly** run on all systems. The java machine is tested rigorously on all released platforms by Sun Microsystems Java team. However, we rigorously test all versions of Automize on the following platforms:

- 1) Windows (both 95 and NT families, including servers)
- 2) MacOSX,
- 3) Linux Redhat

We also provide some releases for other Java 1.3 enabled platforms. We cannot test on all other Java 1.3 platforms, however, since we confirm that it runs perfectly on windows, mac, and linux, we are very confident that it will run on all other java 1.3 platforms.

Debug Log

This is one of the extremely rare software programs that exposes bugs to the users via the Debug Log. The debug log is the single most important information we use to figure out why your tasks fails (either user input error, system or other error or bug). We constantly refine our code when we go through your debug logs. 100% of our code is in error catching loops that write to the debug log. This has helped the program to be extremely stable. By version 5.x, we have cleaned up a lot bugs, from information that filtered into the users 5.x debug log.

Scheduler engine CPU usage

The scheduler engine just sets timers for your tasks. Every task has one timer associated with it. Typically when no tasks are running, the scheduler engine will consume almost very little (~0%) of CPU time. When the task is triggered, the CPU usage will depend on the task.

Scheduler reliability testing

Conditions: Ultra1, Solaris 8.0, 256MB Ram, 166MHz, 2 Tasks (copy & delete) run every 15 seconds for 1 month.

Results: RAM usage plateaus at 60-80 MB during the entire month. CPU usage ~2-5%

Zip Task

Conditions: Win2000, 256MB Ram, 667 MHz, Source folder size = ~1.8 GB, Source files zipped = 20031

Results: Zip time = ~24 minutes, Zip file size = ~1.3GB, Average CPU = ~70-90%

Output:

May 17, 2002 1:23:49 PM

Zip Directories - Zip

Zip session summary:

Number of files in directory = 20031

Number of files passing filename filter = 20031

Number of files passing date modified criteria = 20031

Number of files zipped = 20031

Zip file created : C:\Documents and Settings\melvin\junk\Zip.zip

File Size = 1325729962 bytes

May 17, 2002 1:23:50 PM

Zip Directories - Zip - Exit Code = 0

Copy Task

Conditions: Win2000, 256MB Ram, 667 MHz, Source folder size = ~150 MB, Source files copied = 5667, Subfolders = 419

Results: Copy time = ~5 minutes, Average CPU = ~20-40%.

Output:

May 17, 2002 2:00:10 PM

Copy Files - Copy

Copy session summary:

Number of files in directory = 5667

Number of files passing filename filter = 5667

Number of files passing date modified criteria = 5667

Number of older source files copied (only if 'Ignore file timestamp' option is selected) = 0

Number of source files older than target files (i.e. not copied) = 0

Number of files copied = 5667

May 17, 2002 2:00:11 PM

Copy Files - Copy - Exit Code = 0

Ftp Task

Conditions: Client Win2000, 256MB Ram, 667 MHz. Ftp Server = Solaris Sparc 8.0, 10 Mbps network speed.

Ftp task: Source folder size = 150 MB, Source files transferred = 5642, Subfolders = 419, log transfer or backup options were NOT selected.

Results:

Ftp Get time = ~40 minutes, client CPU usage = varies 5-20%.

Ftp Put time = ~60 minutes, client CPU usage = varies 20-100%.

The Ftp transfer time and %CPU usage will be higher if the log transfer or backup options are selected.

Output:

May 17, 2002 5:58:36 PM

FTP session summary:

Total files = 5642

Files meeting filename criteria (C1) = 5642

Files meeting transfer criteria per date modified & criteria C1 (C2) = 5642

Files meeting transfer criteria based on new modified files & criteria C2 = 5642

Total files downloaded = 5642

May 17, 2002 5:58:37 PM

Ftp - Ftpget - Exit Code = 0

Using Help

There are three options to view the Help file.

- 1) Default java help system. Includes full text search and context sensitive help.
- 2) View HTML help using software browser.
- 3) View HTML help in native browser.

You can set the default Help system in the Settings menu of the software. You can view the latest online version of this help file, by visiting our web site. You can also access the online help, via the Help menu in the front panel.

Front Panel

Table Description

The front panel contains the [All Tasks](#) and [Scheduled Tasks](#) tables. The [All Tasks](#) table contains all the data for each task that you create. The [Scheduled Tasks](#) table contains scheduling data, for the tasks that have been scheduled to run. Each table has an associated toolbar with buttons, which are described in the sections [All Tasks](#) and [Scheduled Tasks](#).

Run Software in the Background

In the right top part of the panel next to the clock, is the Program icon. Click this button, and the user interface will run in the background. The icon will appear on your desktop. Clicking on the icon on your desktop, will open the Front Panel again. You can set the icon to appear at various portions of your screen, by using the "Options" menu , "Icon Location" menu item.

Chain Tasks



Use this button to create a new chain task. Chain tasks can run other tasks in sequence.

Start/Stop Scheduler



If the button is green, the scheduler engine is running. If the button is red, the scheduler is shutdown. This button allows you to start the scheduler engine, or to shutdown the scheduler engine.

Status Bar

At the bottom of the panel is a status bar, which provides information regarding user input.


Task Table

The Tasks table contains all the data for each task. There are 50 parameter columns allowed for each task. Each task may use up only a few of these columns.

You can edit the task using the Edit button. This will load the parameters into the task dialog box correctly, and you can edit the settings there.

To run a task, select the task in the Tasks table, and click the "Run" button. 

To schedule a task, select the task in the Tasks table, and click the "Schedule" button. 

To edit a task, select the task in the Tasks table, and click the "Edit" button. 

To delete a task, select the task in the Tasks table, and click the "Delete" button. 

To duplicate a task, select the task in the Tasks table, and click the "Copy" button. 

Schedule Table

The "Scheduled Tasks" table contains the data for all the scheduled tasks. You can edit the schedule, using the Edit button. This will load the parameters correctly into the schedule dialog box. You can edit the schedule settings, in the schedule dialog box.



To **edit** a task schedule, select the task in the "Scheduled Tasks" table, and hit the "Edit" button.



To **delete** a task schedule, select the task in the "Scheduled Tasks" table, and hit the "Delete" button.



To **run** a task, select the task in the "Scheduled Tasks" table, and hit the "Run" button..



This button resets the next run times for all tasks. This is useful, if some of your tasks have failed with fatal errors, and the engine has stopped scheduling them. After fixing your tasks, click this button to reschedule all tasks. The schedule table will be updated after 5 seconds.




This button updates the Next Run column. **The Next Run column has to be manually updated by the user.**

The "**Run**" column allows you to enable or disable a task schedule.

The "**Frequency**", "**Period**", "**Hour**", "**Minute**", and "**Details**" columns display the scheduling parameters that have been set for this task.

Scheduling Tasks

Tasks can be scheduled by the second, minute, hour, day, week, or month. To schedule a task, select an existing task from the Task table, and click on the schedule button. 

Scheduling by the Second

Tasks can be scheduled by the second. i.e. every $x = 15$ seconds etc. Scheduling tasks at this rate, is very likely to consume a large amount of your computer's time and resources. When you schedule by the second, the task will run every x seconds, from the time the task was created. If you restart the scheduler, the task will run first after x seconds, and then repeat every x seconds.

When scheduling by the Second, the following fields are used: Run Hours, Run Days.

When scheduling by the Second, the following fields are ignored: Run Months, Run Dates, Run Weeks.

Scheduling by the Minute

When you schedule by the minute, two rules apply:

1. If you schedule every 1, 2, 3, 4, 5, 6, 10, 12, 15, 20 or 30 minutes (i.e. divisible by 60), then the task will run at the same time each hour. Example, if you schedule a task to run every 15 minutes, and you set the run 'minute' = 0, it will run at 0, 15, 30 and 45. If you set the run 'minute' = 5, it will run at 5, 20, 35 and 50.
2. If the period is not divisible by 60, the task will run every x minutes. If you restart the scheduler, the task will run first after x minutes, and then repeat every x minutes.

When scheduling by the Minute, the following fields are used: Run Hours, Run Days.

When scheduling by the Minute, the following fields are ignored: Run Months, Run Dates, Run Weeks.

Scheduling by the Hour

When you schedule by the hour, two rules apply:

1. If you schedule every 1, 2, 3, 4, 6, 8, 12, or 24 hours (i.e. divisible by 24), then the task will run at the same time each day. Example, if you schedule a task to run every 6 hours, and you set the run 'minute' = 0, it will run at 12AM, 6AM, 12PM, 6PM everyday.
2. If the period is not divisible by 24, the task will run every x hours. If you restart the scheduler, the task will run first after x hours, and then repeat every x hours, at the minute of the hour you set.

When scheduling by the Hour, the following fields are used: Run Hours, Run Days.

When scheduling by the Hour, the following fields are ignored: Run Months, Run Dates, Run Weeks.

Scheduling by the Day

When you schedule by the day, the task will run on the days selected, at the exact time of the day you set.

When scheduling by the Day, the following fields are used: Run Days, Run Weeks.

When scheduling by the Day, the following fields are ignored: Run Hours, Run Months, Run Dates.

Scheduling by the Week

When you schedule by the week, the task will run every week, on the day selected, and at time of the day you set.

When scheduling by the Week, the following fields are used: Run Days, Run Weeks.

When scheduling by the Week, the following fields are ignored: Run Hours, Run Months, Run Dates.

Scheduling by the Month

When you schedule by the month, the task will run for the months that you have selected. The task will run, on the date of the month, and the time of the day you set. If you require the task to run, on the last day of the month, select the date as "32".

When scheduling by the Month, the following fields are used: Run Months, Run Dates.

When scheduling by the Month, the following fields are ignored: Run Hours, Run Days, and Run Weeks.

Start Date

This field is a Start Date filter. If the calculated next run time, is prior to the Start Date, the next run time is incremented, till it is after the Start Date. This time does not indicate the first run time.

End Date

This field is an End Date filter. If the calculated next run time, is past the End Date, the scheduled is Cancelled. This time does not indicate the Last run time.

Exit Codes

The program attempts to obtain exit codes for your tasks. If it fails to show an exit code, and your task fails, please view the task and output logs, as well as the debug log from the Log menu.

If a task exits normally, without generating any error, Exit Code = 0. For many tasks, which involve monitors or file downloads, the Exit Code = -100 or -101, if the tasks monitoring or download criteria are met.

Task Type	Error description
CopyFile	EC = 1, File copy failed, or delete failed after rename/move options EC = 2, Invalid target directory EC = 3, Invalid source directory EC = 100, Invalid task data format
PrintFile	EC=0 always. No Exit codes provided. Please view output log for error information.
DeleteFile	EC = 1, File delete failed EC = 2, Invalid local directory EC = 100, Invalid task data format
Text Search	EC = -100, no error, search string(s) found in file(s) EC = 0, no error, search string(s) NOT found in file(s) EC = 1, task failed, no other details, check logs EC = 2, Invalid results directory EC = 3, Invalid source directory EC = 4, Error occurred while reading source file EC = 5, Error occurred while writing to results file EC = 100, Invalid task data format
Command	EC != 0 , System/program generated EC = 100, Invalid task data format EC = 101, Unknown (check log) EC = 200, Wait time exceeded
WinCommand	EC != 0 , System/program generated EC = 1, WinCommand.exe Failed to start process EC = 2, process did not allow WinCommand.exe to wait on it EC = 3, process abandoned without providing feedback EC = 10, could not open or access WinCommand.exe input data file EC = 100, Invalid task data format EC = 101, Unknown (check log) EC = 258, Wait time exceeded
Ftp	EC = -100, no error, file changes were detected, files were transferred EC = 0, no error, no file changes were detected, no files were transferred EC = 1, Failed to transfer/delete file(s) EC = 2, Failed to connect to server EC = 3, Invalid user/password EC = 4, Invalid remote directory EC = 5, Failed to set Binary EC = 10, Invalid local directory EC = 100, Invalid task data format

Ping	<p>EC = 0, no error, connection accepted</p> <p>EC = 1, connection denied or timed out</p> <p>EC = 5, ping not attempted, time since last Ping < 5 minutes.</p> <p>EC = 100, Invalid task data format</p>
Telnet	<p>No exit code information except connection status.</p> <p>EC = 0, connected successfully</p> <p>EC = 1, failed to connect</p>
Telnet(Adv)	<p>No exit code information except connection status.</p> <p>EC = 0, connected successfully</p> <p>EC = 1, failed to connect</p>
Email	<p>EC = 1, Email failed, check debug and/or output logs</p> <p>EC = 100, Invalid task data format</p>
Bulk Email	<p>EC = 1, Task failed, check debug and/or output logs</p> <p>EC = 100, Invalid task data format</p>
Check Email	<p>EC = -100, no errors, messages were downloaded to the mail folder</p> <p>EC = 0, no error, no messages satisfied conditions for downloading</p> <p>EC = 1, Task failed, check debug and/or output logs</p> <p>EC = 2, Failed to connect to email server</p> <p>EC = 3, Failed to open the default INBOX folder</p> <p>EC = 4, Failed to obtain a list of messages in your INBOX folder</p> <p>EC = 100, Invalid task data format</p>
Web	<p>EC = 1, Download failed</p> <p>EC = 100, Invalid task data format</p>
Url Monitor	<p>EC = 0, no error, no files downloaded</p> <p>EC = 1, no files downloaded, errors detected</p> <p>EC = 100, Invalid task data format</p> <p>EC = 105, Failed to create sub directory</p> <p>EC = -100, no error, Url change(s) detected, file(s) downloaded</p> <p>EC = -101, Url change(s) detected, file(s) downloaded. Some error(s) detected, but these errors are generally due to task trying to download invalid protocol files or directory listings.</p>
Database	<p>EC = -100, no error, database change detected during Select statement</p> <p>EC = 0, no error, database change was NOT detected during Select</p> <p>EC = 1, failed to connect to Database</p> <p>EC = 2, SQL exception</p> <p>EC = 3, other exception within SQL loop</p> <p>EC = 4, error writing select statement data to file</p> <p>EC = 100, Invalid task data format</p>
Dir Change	<p>EC = -100, no error, Dir/File change was detected</p> <p>EC = 0, no error, Dir/File change was NOT detected</p> <p>EC = 2, Invalid Dir/file</p> <p>EC = 201, Other unknown error (check log)</p>
File Monitor	<p>EC = -100, no error, Dir/File change was detected</p> <p>EC = 0, no error, Dir/File change was NOT detected</p> <p>EC = 3, Invalid Dir/file</p> <p>EC = 1, Other unknown error (check log)</p> <p>EC = 100, Invalid task data format</p>
Dir Monitor	<p>EC = -100, no error, Dir/File change was detected</p> <p>EC = 0, no error, Dir/File change was NOT detected</p> <p>EC = 2, Invalid Dir/file</p> <p>EC = 201, Other unknown error (check log)</p>

Variable Monitor	<p>EC = -100, no error, Variable comparison criteria was met</p> <p>EC = 0, no error, Variable comparison criteria was NOT met</p> <p>EC = 2, Variable Name was not found in the users variables file (..Automize/userVariables) or in default java system variables.</p> <p>EC = 201, Other unknown error (check log)</p>
Chain	<p>Task EC = 102, Task wait time exceeded</p> <p>Chain EC = step at which failed, Chain failed</p> <p>Chain EC = 100, Invalid task data format</p> <p>Chain EC = 101, Task not found in Task Table</p>
Ftp Monitor	<p>EC = -100, no error, file changes were detected</p> <p>EC = 0, no error, no file changes were detected</p> <p>EC = 1, Error detected, see output log for details</p> <p>EC = 2, Failed to connect to server</p> <p>EC = 3, Invalid user/password</p> <p>EC = 4, Invalid remote directory</p> <p>EC = 100, Invalid task data format</p>
Ftp Command	<p>EC = 0, no error</p> <p>EC = 1, see output log for details</p> <p>EC = 4, invalid custom command</p> <p>EC = 100, Invalid task data format</p>
Zip File	<p>EC = 1, File copy failed, no other information, check logs</p> <p>EC = 2, Invalid target directory for zip file</p> <p>EC = 3, Invalid source directory</p> <p>EC = 100, Invalid task data format</p>

Log Files

Task data is logged into log files on your disk. The logs are available via the Logs menu. The log files are automatically trimmed back to 0%, 25%, or 50%, when they exceed the maximum size limit set by user.

Output log

This is the most used log. This log displays any output from your tasks. The data includes run times, any output data, and error data and exit codes and end times.

Debug log

Our software is one of the extremely rare software programs, that can expose bugs to the users via the Debug Log. This log displays any error from your tasks, and the exact location in our program code, where this error occurred. This log provides the most important information, and helps us quickly determine the cause of your task error. We can determine if it is a user input problem, system or other problem, or a bug. Almost ~100% of our code is in error catching blocks. We keep refining our code, when we analyze your debug log. This has helped the program to be extremely stable.

Activity log

This log displays a list of tasks run times, and startup, and shutdown times.

Task logs

Each individual task that you create, will output data into task log files. The log files have the same name as the task title. The data that is displayed in these logs include run times, any output data, error data, and exit codes.

Filename Format for Append To Filename option

Many of the tasks have the 'Append to filename' option. You can append Date/Time to the filename. You can also add your own Custom Text to your filename. You can also set the format that you want the Date/Time. Use the options indicated below, in Settings Menu / FilenameFormat dialog.

Appending Custom Text to your filename:

You can add your own Custom Text to your filename, according to the rules below:

- 1) Text will add 'Text' to the end of the filename (after extension).
- 2) P1::Text will add 'Text' before the filename
- 3) P2::Text will add 'Text' before the extension
- 4) P3::Text will add 'Text' at the end (after extension if an extension exists)
- 5) P4:: will add the Date to filename (before extension if an extension exists)
- 6) P5:: will add the Date/Time to filename (before extension if an extension exists)
- 7) You can add multiple Text to your filename using + after each rule used
- 8) You can also enter task or system variables to be added to the filename, except for the P4:: & P5:: formats, which already add the date or date/time.

IMPORTANT: P1:: , P2:: etc., should be entered exactly as shown (case sensitive)

Examples:

Filename	Custom String Entered	Resulting Filename
test.txt	-java-\$%DATE::ppddy%\$	test.txt-java-012704
test.txt	P1::pre_	pre_test.txt
test.txt	P2::_beforeExt	test_beforeExt.txt
test.txt	P2::_\$%DATE::ppddy%\$	test_012704.txt
test.txt	P3::_afterExt-\$%DATE::ppddy%\$	test.txt_afterExt-012704
test.txt	P2::_beforeExt_+P4::	test_beforeExt_01-27-04.txt
test.txt	P1::pre_+P5::	pre_test01-27-04_12:32.txt

Appending Date or Date/Time

If you select 'Date', the current date is appended to the filename. If you select 'Date/Time', the current date and time is appended to the filename. This is useful for archiving purposes. The date and month are always 2 digit numbers. If time is appended, the hour and minute are always 2 digit numbers. The standard date time separators like / or : are not used. These separators cannot be used in pathnames or filenames, on some operating systems. Hence we chose -, _ , or none as the separators.

Date Format Sequence:

This allows you to select the sequence of date, month, and year.

Year Format:

You can set a 2 or 4 digit year, or choose not to have the year appended to your filename.

Date Time Separator:

This sets the separator between the date and time parts, if the 'Date/Time' 'Append to Filename' option is selected.

Date Component Separator:

This sets the separator between the date, month and year.

Time Component Separator:

This sets the separator between the hour and the minute fields, and between the minute and the second fields.

Include Seconds:

Select this option to append the second's field to the time.

Filename Selection

We provide a custom wildcard system. This system provides more flexibility than either the DOS or Unix wildcard systems. You can select multiple files from a directory using the following rules:

- 1) Multiple sets of filenames can be selected using ^.
- 2) Each filename can contain 1 or more string tokens, separated by *.
- 3) Each * also stands for one character (or more).
- 4) Filenames containing all individual tokens entered, in the correct sequence, will be selected.
- 5) The index of an individual token in the filename selected cannot be less than the index of that token in the filename entered.
- 6) The length of the filename selected, should always equal or exceed, the length of the filename entered.
- 7) If * (or multiple *) is at the end, **it is ignored**. However, the length of the filename selected, will equal or exceed, the length of the filename entered.
- 8) For exact filename search, use # at start of filename. This ensures only one file with the exact name, will be selected.
- 9) To exclude filename(s), use ! at the start of filename. The exclusion filename **should be followed by at least one other token** separated by ^. The exclusion filename(s) should also be listed before any normal filename. Examples: !.zip^* or !.zip^.txt or !.txt,!.zip,.htm,.gif

Consider a directory with the following files:

junk, test, temp, temp1, temp.txt, temp.htm, Test, Temp.txt, teeempju.txt, activityLog.txt, debugLog.txt, test.zip, test.gif

Filename entered	Files Selected
	All files selected
*	All files selected
te	test, temp, temp1, temp.txt, temp.htm,teeempju.txt
ju	junk,teeempju.txt
T	Test, Temp.txt
.txt	temp.txt, Temp.txt, teeempju.txt, activityLog.txt, debugLog.txt
te*htm	temp.htm
te*emp	teeemp.txt
te*htm^debug	temp.htm, debugLog.txt
te*htm^te*emp^debug	temp.htm, teeemp.txt, debugLog.txt
temp*.txt	No files selected (temp.txt is not selected per rule 5)

*****	teeempju.txt, activityLog.txt, debugLog.txt (per rule 6)
est**	No files selected (test, Test are not selected per rule 6)
est*	test, Test (per rule 6 and 7)
#temp	temp (per rule 8)
!.zip^*	All files except test.zip (per rule 9)
!.txt^temp	temp,temp1,temp.htm (per rule 9)

Scheduler Settings

Engine server port

The software contains two separate programs. An administrator user interface, and a scheduler engine program, which runs in the background. The administrator user interface sends data to the Scheduler Engine Server Socket, which is listening by default on port **1965**. It is very unlikely that you will need to change this default port value. This default value will need to be changed, **ONLY** if another server is running on your system, and is listening on this port. If you change the default port value, **BOTH** the administrator user interface, and scheduler engine program need to be restarted. This Scheduler engine running in the background, allows for use as Service on Unix, Linux, Solaris, Windows NT , 2000, XP. Using Sockets to communicate, allows for future use of a Remote Administrator, to control the scheduler engine.

Launch wait time

This is the maximum time in seconds, that the user interface waits, for the Scheduler engine to respond on startup. If your system is slow, or you have many tasks and schedules, the scheduler engine may take longer than the default (60 seconds) to startup. In this case, you may get a message saying that the scheduler engine has failed to startup. In reality, the scheduler engine may have started, but only after 60 + seconds. If you receive this message often, then increase this setting to 120 seconds.

Scheduler sleep time (ms)

Every time a schedule is triggered, the scheduler will set the next schedule and run the task. Beginning in version 5.4_3, you can set the scheduler to wait for a few milliseconds (1000 milliseconds = 1 second), before setting the next schedule. This helps to prevent multiple triggers, on some older systems. These older systems may return faulty values of the current time. In versions 5.0 - 5.4_2, this value was fixed at 1 millisecond. Starting in version 5.4_3, this value can be set by the user. The default is now 100 milliseconds, but you can choose to set it between 0-60000 milliseconds (0-60 second). If you set a sleep time of 5000 milliseconds (5 seconds), the task will start running 5 seconds after the schedule is triggered.

Time Zone

The scheduler engine automatically detects the system time zone settings, indicated by the 'default' value. If your schedules are 1 hour off, you may have to reset the time zone to the correct value.

Thread Priority

All tasks are multithreaded in version 6.x. Every task runs as a separate thread in the engine. You can set the thread priority to be minimum, normal, or maximum. It is recommended that you use either minimum or normal (default) priority. Maximum priority is NOT recommended.

Check for multiple triggers within 30 seconds

In version 5.x, two users had reported multiple task instances (2-6 instances), being triggered at the same time. The cause of this multiple triggers is unknown. Select this option, if you happen to suffer from this multiple trigger problem. This option does not apply, when the scheduling period is by 'Second'. If you select this option, and two instances of the same task are triggered within 30 seconds of each other, then the second instance is cancelled.

Scheduling Error

In version 5.x and earlier, when a scheduling error occurred, the task schedule was cancelled, and then rescheduled. In 6.x, more corrective options are provided.

- 1) No corrective action is taken.
- 2) Reschedule only the task schedule, that led to the error.
- 3) Reschedule all the task schedules.
- 4) Restart the scheduler engine. In this case, all pending schedules are cancelled, and a new instance of the scheduler engine is launched. However, if any task is still running in the current instance of the engine, it will continue to run. The current instance of the engine will shut down, after all running tasks have finished, or when the maximum wait time (default = 60 minutes) is reached.

Wait for tasks to Finish before Engine shuts down

When the engine is shut down, all pending schedules are cancelled. However, if any task is running in the current instance of the engine, it will continue to run. The current instance of the engine will shut down, after all running tasks have finished, or when the maximum wait time (default = 60 minutes) is reached.

Maximum wait time before engine stops

When the engine is stopped, the current instance of the engine will shut down, after all running tasks have finished, or when this maximum wait time (default = 60 minutes) is reached. You can change this default value as required.

Unix Service

This software contains two separate programs. An administrator user interface, and a scheduler engine program, which runs in the background. **By default, the scheduler engine in the background, is not started with the 'nohup' command.** Hence, when you logout, the scheduler may shutdown. By using the 'nohup' command to launch the scheduler engine, the scheduler engine will keep running after user logout.

Restart engine using launch code

You can re-launch the engine using the '**Restart engine using launch code**' button. This automatically shuts down the current instance of scheduler engine. Then, the engine is restarted, using the new 'nohup' code.

Recommended nohup launch code

The recommended command to launch the engine using 'nohup', is shown in the text box. You can edit it as you wish.

Save code and always launch scheduler engine using this launch code

Select this checkbox, if you wish to always use nohup, to launch the scheduler engine.

Important Notes

- 1) If you are re-launching the scheduler engine, through a script or from a terminal, make sure that you shut down the scheduler engine, through the user interface first.
- 2) If launching Engine from commandline/script, start dir should be software install folder. Please 'cd' to the software install folder first.

Default Scheduler Engine Launch Code

If you used the install with JVM:

```
jre/bin/java -cp ./inputs:images:activation.jar:mail.jar:NetComponents.jar:jclasses.jar JSEngine
```

or if you used the install without JVM and java is in your path

```
java -cp ./inputs:images:activation.jar:mail.jar:NetComponents.jar:jclasses.jar JSEngine
```

Note: On some systems, like Linux Redhat 7.0, 'nohup' is not needed. The scheduler engine may keep running on logout, even if launched without 'nohup'. On Solaris 8.0, the scheduler engine will shut down if 'nohup' is not used.

Archive Logs

This administrative task allows you to archive the Output, Activity and all Tasklogs. To create an Archive Log task, hit the Save button in the dialog box. An entry will be added into the 'All Tasks' table. You can then schedule this task.

Target Directory

Enter the full path name of the directory, where you need to put the log files. Note that the path is case sensitive on unix systems.

Place files in new subdirectory

If you select "None", no new subdirectory is created within the Target Directory.

If you select Date, a new subdirectory based on the date, is created within the Target Directory. All files are placed in this directory.

If you select Date, a new subdirectory based on date and time, is created within the Target Directory. All files are placed in this directory.

Append to Filename

To archive each log file, based on current time/date, [choose the append file option.](#)

Auto Backup

This administrative task allows you to backup all the user settings, task table and schedule table data. To create an Auto Backup task, hit the Save button in the dialog box. An entry will be added into the 'All Tasks' table. You can then schedule this task.

Target Directory

Enter the full path name of the backup directory. Note that the path is case sensitive on unix systems.

Place files in new subdirectory

If you select "None", no new subdirectory is created within the Target Directory.

If you select Date, a new subdirectory based on the date is created within the Target Directory. All files are placed in this directory.

If you select Date, a new subdirectory based on date and time is created within the Target Directory. All files are placed in this directory.

Bulk Email

The email task allows you to send email automatically to multiple recipients in sequence. This allows you to hide the identity of recipients from each other. This also allows you to send email to a large audience, without manually typing in the email addresses, using an email list file. You need to specify the recipients in a text file. Each address has to be on a new line. To create a bulk email task, select the "Create Task" menu and then the "Bulk Email" menu item.

Email Server Profile

Select the profile name for the email server you need to connect to. You should have previously created an email profile, using the Email Profiles menu item.

Recipient list file and additional parameters for mail merge

Select the recipient list file. The file should contain the list of email addresses. One email address per line. Additionally, if you wish to pass multiple parameters for each email address, enter them after the email address separated by ^. Then in any field (subject, message body, headers or attachment paths) you can specify a parameter using %%1, %%2 etc..

Example of Recipient list File:

```
rob@test.com^Rob Gibson^London^60
```

```
jane@test.com^Jane Doe^New York^90
```

In your email, you can access the Name using %%1

In your email, you can access the City using %%2

You can also access the recipient email address using %%0

Example Body:

Dear %%1, The temperature in %%2 is %%3 F. Your email address is %%0.

This would result in the following body sent to rob@test.com:

Dear Rob Gibson, The temperature in London is 60 F. Your email address is rob@test.com.

Subject\$

Enter the subject. This field also supports [dynamic variables](#). For example, you could automatically parse the current date or computer IP address into the subject

Delay

This is the time interval in seconds between successive emails. Normally a value between 5-60 seconds should be used.

Message Body\$

1) Enter the message body directly into the text field

2) This field also supports [dynamic variables](#). For example, you could automatically parse the current

date or computer name into the message

3) Or, You can let the software dynamically parse the message text from a file on your system using the following format:

\$\$\$[message_file_path]

for example on windows, if the message file is c:\data\message.txt, use:

\$\$\$c:\data\message.txt

Adding Attachments

1) To add an attachment, type in the full path name of the file, into the Attachment Text Box. You can also use the "Browse" button to select the file. The format for the attachments is: AbsoluteFilePath1, AbsoluteFilePath2,.....

Example: c:\data\test.txt,c:\temp\data.htm

2) You can also specify multiple filenames, in multiple directories, by using the following format:

[DIR]c:\your_dir[FILE]name_filter,.....

Enter the name_filter for the files you wish to transfer, [using the following wildcard rules](#)

Example: to attach all files that end with .txt from the c:\temp folder, use:

[DIR]c:\temp[FILE].txt

3) This field also supports [dynamic variables](#).

Example:

To attach all files in the c:\temp folder, which has the current date (example: 06-21-02) in their filename, use:

[DIR]c:\temp[FILE]\$%DATE::pp-dd-yy%\$

See [dynamic variables](#) for more details on formatting the date

Headers

You can specify headers to add to the email. This will allow you to add mail priority, or other specific flags. An email server can decide not to display all the headers you specify, or may not understand the headers you specify. For example, if you attempt to specify Email priority level, an email server may accept any, or all, or none of the following flags: X-Priority, X-MSMail-Priority, Importance, and Priority.

To specify headers, use the following syntax:

header1=value1^header2=value2^header3=value3

Header Examples:

X-Priority=1^ Priority=Urgent

Importance=high

Reply-To=test@test.com

Command

This feature allows you to run executable programs. To run shell commands, batch/script files, you will need to launch the appropriate shell as an executable. Then use the shell to run the command.

Command Line \$

Executables: If you want to simply launch an executable, browse to it, or enter it manually. Example:

```
c:\winnt\notepad.exe
```

If the path to your executable includes spaces, enter the path to the executable in quotes. Example:

```
"c:\program files\myexe.exe"
```

If you want to enter a commandline parameter, to pass to your executable, enter a space between the path and the parameter. Leave spaces between any subsequent parameters. Example:

```
"c:\program files\myexe.exe" par1 par2
```

For Shell commands, and batch or script files: the syntax required vary depending on operating systems. [Examples](#) of batch commands for [Win 95/98](#), [WinNT/2000pro](#), [Unix](#), and [Mac OS X](#) are listed at the end of this section.

This field also supports [dynamic variables](#). For example, to launch a batch file, and pass it the current time, use:

```
c:\data\test.bat %DATE::hh:mm:ss%
```

Working Directory

The working directory is required for your program, command or script, to run correctly. This is particular true if the program or script uses relative paths to access and output to files.

Environment Variables \$

This is an array of strings, each element of which has environment variable settings in format name=value. You should separate each name=value pair by a delimiter character. You can choose the delimiter to separate each element. The default delimiter is ^. If any of your variable name=value pairs contains the ^ character, then you should set another character as your delimiter. The delimiter cannot be "=" or any other character that appears in your name=value pairs.

Example of correctly formatted environment variables on windows systems:

```
PATH=c:\test;c:\test2;c:\test3^TEMP=c:\temp
```

This field also supports [dynamic variables](#). For example, to launch a batch file and set the variable 'CUSTOM_DATE', use:

```
CUSTOM_DATE=%DATE:::Qqq dd, YY HH:mm:ss am_pm%
```

This will set the variable CUSTOM_DATE in the batch file to: Aug 21, 2002 10:21:23 AM etc..

Termination Time

The termination time is the maximum time that your program or script can run. The task will terminate your program, if it is still running, after the maximum allowed time. Enter a value = 0, if you do not want to terminate your program. In this case, the task will wait indefinitely, for your task to finish.

Polling interval

While your program or script is running, the task continuously polls your program/script. The task will check if the program has completed. The task will obtain the Exit Code value, when your program exits. You can select the polling interval (default = 15 seconds). The minimum interval is 1 second. We recommend using 15 seconds.

Command line Examples

Win 95/98

1) Executable programs:

Command line = c:\windows\notepad.exe

Working Directory = c:\windows

2) Batch files (uses command.com, located usually in c:\windows or in c:\)

Command line = c:\windows\command.com /c c:\test\test.bat

Working Directory = c:\test

NOTE: you can also try and run batch files, just like an executable, depending on your system setup.

Command line = c:\test\test.bat

3) Shell commands (uses command.com, located usually in c:\windows or in c:\)

Command line = c:\windows\command.com /c set c:\test\set.txt

Working Directory = c:\windows

(this example will output all your environment variables to the file set.txt)

WinNT/2000pro

1) Executable programs:

Command line = c:\winnt\notepad.exe

Working Directory = c:\winnt

2) Batch files (uses cmd.exe, located usually in c:\winnt or c:\winnt\system32)

Command line = c:\winnt\system32\cmd.exe /c c:\test\test.bat

Working Directory = c:\test

NOTE: you can also try and run batch files, just like an executable, depending on your system setup.

Command line = c:\test\test.bat

3) Shell commands (uses cmd.exe, located usually in c:\winnt or c:\winnt\system32)

Command line = c:\winnt\system32\cmd.exe /c set c:\test\set.txt

Working Directory = c:\winnt\system32

(this example will output all your environment variables to the file set.txt)

Unix (Solaris, Linux, MacOSX or other)

To run shell commands or scripts, you will need to launch the appropriate shell as an executable. Then use the shell to run the command.

Example to run the script 'myscript' from the /home/name directory, using the 'sh' command interpreter:

Commandline = sh myscript

Working Dir = /home/name

To see the options available to run your script, use man sh, man csh, or man bash.

Another option to run a shell script or command, is to use the Telnet Task. Using the Telnet task, login to the local system, 'cd' to the desired director, using a first telnet command. Then run the shell script or command, using a second telnet command.

Mac OS X

1. **AppleScripts:** Use the 'osascript' command to launch your AppleScripts

Example:

Commandline = /usr/bin/osascript /user/Name/YourScript.scpt

Working Dir = /user/Name

2. **Launch Applications or files:** You can also use the 'open' command in MacOS X, to open a document or application

Example:

Command Line = open Stickies.app


Working Dir = /Applications

3. **Commandline:** To run shell commands or scripts, please see the section on Unix above.

Chains

This feature allows you to run multiple tasks in sequence. You can run programs, scripts, email, web, ftp and most other tasks from the Chain. If a certain task fails, you can choose to stop the Chain. You can also skip ahead to another task in the chain.

Creating a Chain

From the front panel click on the Chain button .

Adding Tasks to your Chain

The table on the upper right, displays the tasks you can add to your Chain.

Select the task you need to add and choose your task options.

Hit the "Add Task" button. Your task will be added to the table, on the bottom of the window.

Email Notification of tasks in the chain

By default, only the Chain itself can trigger email notification, based on exit code. The tasks in a chain, do not trigger email notification, even if they satisfy an email notification profile. You should select this option, in order to allow tasks in the chain to trigger email notification. The chain will move on to the next step, only after the email notification has been completed.


Starting Chain at specified step

You can specify a step to start at, by entering the step number, or step title in this field. By default, the chain will start at step 1, if this field is left blank.

Deleting Tasks from your Chain

Select the task in the bottom table and hit the delete button. .

Editing Tasks in your Chain

Select the task in the bottom table and hit the edit button.  The task parameters will be loaded into the window. You can then make any changes you require and hit the "Update Task" button.

Positioning a Task within your Chain

Select the task in the bottom table and use the move up  or move down  buttons as required.

Enable Step

You can enable or disable a step. If a step is disabled, the task will move on to the next step.

Skip to Step options

You can choose to skip to certain step in the chain, if a certain exit code criteria is met. If you wish to stop the chain, because a certain exit code criteria is met, enter a Skip to Step value, which is greater than the total steps in the chain.

Skip to Step (or Title)

- 1) You can enter either the step number in the chain, or the task title. This has to be a valid task title in the chain list.
- 2) You can also run multiple tasks simultaneously using the following rules:
 - a) Separate each Task Title by &. Example: TaskTitle1&TaskTitle2&TaskTitle3 (no spaces allowed)
 - b) You can only enter Task Titles, NOT step numbers.
 - c) TaskTitle1 has to be a valid Task Title in the chain list
 - d) TaskTitle1 CAN NOT be the last step in the chain list. You can add a dummy Echo task as the last step in the chain, if TaskTitle1 happens to be your last step.
 - e) TaskTitle2, TaskTitle3 etc.. DO NOT have to be in the chain list. They have to be valid Tasks in the main Task Table.
 - f) In this simultaneous mode, all tasks (TaskTitle2, TaskTitle3.. etc..) will be triggered simultaneously. The chain will then continue normally at TaskTitle1.

Rerun Options

You can choose to rerun the task, if it meets certain exit code criteria. You can also set the maximum number of times to rerun a task. The task will be rerun, while the rerun criteria is valid, until the maximum rerun count has been met.

Maximum Wait Time

This is the maximum time (**T1**) the Chain will wait for your task to finish. [*Please note that the Chain will not kill or terminate your task because it is unsafe to do so.*](#) It just waits for your task to complete normally. If the task does not complete normally in the allotted time, the chain will assume that the task has failed. The email, web, ftp and most other tasks will terminate normally.

For the **DirChange** and **Command** tasks, please ensure that you can Terminate the task within the Task settings itself. Both these tasks have a safe Termination time (**T2**) that can be set within the task. Please set **T1 T2**. If you have set the Command or DirChange task to rerun on failure, please set the **T1 (T2 +Td)*n**, where **Td** is the delay between reruns, and **n** is the number of reruns.

Why not just Kill a task when the wait time is exceeded

Short answer from us: In the chain, each task is run as a thread. Killing a thread is unsafe, and may corrupt your system hours or days in the future.

Long answer from Java documentation: Because it is inherently unsafe. Stopping a thread causes it to unlock all the monitors that were locked. (The monitors are unlocked as the Thread Death exception propagates up the stack.) If any of the objects previously protected by these monitors were in an inconsistent state, other threads may now view these objects in an inconsistent state. Such objects are said to be damaged. When threads operate on damaged objects, arbitrary behavior can result. This behavior may be subtle and difficult to detect, or it may be pronounced. Unlike other unchecked

exceptions, Thread Death kills threads silently; thus, the user has no warning that his program may be corrupted. The corruption can manifest itself at any time after the actual damage occurs, even hours or days in the future.

Copy Files

The CopyFile task allows you to copy file(s) from a directory or, an entire directory tree. The 'move' option deletes the original files, after a successful copy operation. The 'rename' option renames the original file(s) to the new name(s).

Source Directory

Enter the full path name of the directory where your source files exists. Note that the path is case sensitive for unix Ftp servers.

Target Directory

Enter the full path name of the directory where you need to put the target files. Note that the path is case sensitive on unix systems.

Filename\$

For the Copy and Move options only, enter the name filter for the files you wish to copy [using the following wildcard rules](#) . This field also supports [dynamic variables](#). Using dynamic variables, you can select a file based on current date/time in addition to the normal wildcard search.

For the Rename option, the above wildcard options only apply for the file to pass the filename filter. If the file passes the filename filter, then its newly renamed name will depend on the Rename Filename field.

Rename Filename

The rename filename field entry is dependent on the filename field entry.

- 1) You should specify matching tokens in the Filename and Rename Filename field.
- 2) Multiple tokens should be separated by ^ .
- 3) If the filename field has more tokens than the rename filename field, then the first token of the rename filename field is used only.
- 4) If the rename filename field is blank, this option will act similar to the move option.

Consider the following files in folder = test.txt, temp.zip, temptest.txt

Filename field	Rename Filename Field	Rename results
.txt	.doc	test.doc, temptest.doc
.txt^.zip	.doc^.jar	test.doc, temp.jar, temptest.doc
temp^test	bad	bad.txt, bad.zip, badbad.txt
te	bad	badst.txt, badmp.zip, badmpbadst.txt
temp^test^.zip	bad^.txt	bad.txt, bad.bad, badbad.zip

.txt	(left blank)	test.txt, temptest.txt (same as move option)
------	--------------	--

Please note that, when using the rename option, some dynamic variables like `$(....):FileNames%`, will give unpredictable results. This is because, more than 1 filename is returned by this variable. i.e. There is no way to match the filename tokens, with the rename filename tokens.

Place files in new subdirectory

If you select "None", no new subdirectory is created within the Target Directory.

If you select Date, a new subdirectory based on the date is created within the Target Directory. All files are placed in this directory.

If you select Date, a new subdirectory based on date and time is created within the Target Directory. All files are placed in this directory.

Include subdirectories

If you select this option, all subdirectories and files within the Source Directory are copied to the Target Directory. If a subdirectory does not exist within the Target directory, then it is automatically created.

Append to Filename

If you are regularly copying a file and wish to save each copy to a different filename based on current time/date, [choose the append file option which best suits your copying frequency and file naming needs.](#)

Date Filter

If you need to filter files based on file modified date, select this option. For the between option, enter 2 values separated by '-' (2-4 etc..). Examples:

Older than 5 Minute, Newer than 2 Day, Between 3-5 Day, Between 1-4 Hour

Delete empty subfolders after Move

The move option first moves all files that meet the filename criteria, and other criteria. By default, empty subfolders are not deleted, after all files are moved out. This is done by design. Many production systems may need these empty subfolders to remain in place. This is for other processes, to move new files, into these empty subfolders.

If you need to delete empty subfolders, after the Files have been moved, select this option. Please note that all empty subfolders, in the source folder, will be deleted. This is true even, if there were no files in those subfolders, at the start of the move option.

Check Email

This task allows you to download email messages, from your email server, to a user-specified folder. Only messages within your INBOX folder are searched. You can specify conditions based on date, sender or subject. Only messages that match your specified conditions, will be downloaded. Any attachments are also downloaded.

Conditions

Condition 1 is required. Condition 2 is optional.

For **date** comparison, enter the number of days required. For example, to download all messages that have been received in the last 3 days, enter **3** in the text box.

For conditions based on **subject** or **sender**, you have four choices. **The comparisons are always case sensitive.**

and/or

Choose **and**, if both Condition 1, and Condition 2 should be satisfied. Choose **or**, if either Condition 1, or Condition 2 can be satisfied.

Email Folder

Please enter a valid folder path, on your local system. All emails will be downloaded to this folder. The format of the email names is the name_x_ , where name is your Check Email task title, and x is the message number that has been downloaded. For example, if your task title is "emailTest", and 3 messages were downloaded, the 3 email files will be emailTest_0_ , emailTest_1_ , and emailTest_2_ . All attachments will also be downloaded to this folder, and will maintain their original names.

[Append to filename](#)

You can save downloaded messages and attachments, to a different filename, based on [date, or date/time, or custom code you set.](#)

Email Server Type

Please consult your email service providers documentation, for information on your email server type. POP3 is currently more popular. IMAP is the newer protocol, and many email servers will move to IMAP in the future.

Email Server Name

This is the most difficult parameter to get right in this task. **Please consult** your network administrator, or your internet service provider's documentation. Look for POP3, IMAP, or incoming email server name. The exact name is required. The name is case sensitive. URL's are not allowed. If you use email clients like Netscape, Outlook, Lotus, or Eudora, please copy this setting from the software itself.

Email Server Port

The default POP3 email port is usually 110.

Username

The username is case sensitive. No white space at the end.

Password

The password is case sensitive. No white space at the end.

Database SQL

You should test your database connection and SQL statements using the [Database Test](#) utility, before you can make or schedule SQL tasks.

SQL statement

Enter your SQL statement here. It can be a select, insert, update, or delete statement. This field also supports [dynamic variables](#).

Directory to place Results

Enter the full path name of the directory where you need to put the SQL query results. Note that the path is case sensitive on unix systems.

Results Filename

Enter the filename for query results. If no filename is entered, a default filename of your Task Title is used. This field also supports [dynamic variables](#). Using dynamic variables, you can save the file to a customized name based on date/time or username etc...

Append to Filename

To save each query to a different filename, based on current time/date, [choose an append file option](#).

Database JDBC driver

For your database, you need to obtain the JDBC driver from your database vendor or other 3rd party vendor. Many of the database vendors have free JDBC drivers available at their web site. The JDBC driver is generally in the form of a .zip or .jar file. **You need to place the JDBC driver in the following folder, depending on your installation method. Then restart the engine and the user interface.**

For installs including JVM : ...Automize/jre/lib/ext

For installs without JVM : ...jre/1.3/lib/ext or JDK1.3/jre/lib/ext etc.. depending on your system

Driver class name

Enter the class name for the driver. This information will be available in the driver documentation. For Microsoft Access database, simply enter "access". Also, the JDBC driver required for MSAccess is included within java itself.

Database Url

Enter the Url connection information. This information will be available with the Java JDBC driver documentation.

The Url format will vary with vendor. The format includes the server name, port, database name, username, and password.

Authorization

If your database requires authentication, then select this option and enter your username and password.

Username

Enter your username if required for database access. It is case sensitive.

Password

Enter your password if required for database access. It is case sensitive.

Directory Change

The Directory Change task monitor polls a single directory (or file) for changes. The Directory Change task is multithreaded, and many Directory Change tasks can run simultaneously. The Directory Change task can be used in three ways:

- 1) If a change is detected in this task, you can run another task and exit.
- 2) You can set this task to run as a daemon, which continuously polls the directory/file for changes, and keeps triggering another task. This is very useful in production environments, where files are continuously moving into a folder. For example, when a change is detected, you can continuously launch a file processing script. Then continue polling for changes.
- 3) The Directory Change task can also be embedded within a Chain task, and is useful for conditional task processing. If you want to use this task in a chain for conditional processing, select 'None' as the 'Task To Run'. If the criteria is satisfied, then an exit code = -100 is thrown. Else, an exit code = 0 is thrown.

IMPORTANT: This task only detects changes while the task is running. So, if you start/restart the engine at 10:00 AM, and if your next DirChange task is at 10:30 AM, any directory changes between 10:00 AM - 10:30 AM, will go undetected. This task is best used on servers, where you do not have to restart the engine often.

Directory

Enter the directory or file you need to monitor.

Task To Run

Select the desired task to run if any of the selected criteria are met. If you want to use this task in a chain for conditional processing, select 'None'. You can also select multiple tasks, to run in sequence or, to run simultaneously. To run tasks in sequence, use taskTitle1|taskTitle2|taskTitle3. To run tasks simultaneously, use taskTitle1&taskTitle2&taskTitle3.

Maximum wait time

Set the maximum allowed time to wait for a change to occur (**T1**).

Polling Interval

Enter the polling interval in seconds. The default value of 60 seconds is recommended. You can choose a smaller interval if you require.

Use as Daemon

Select this option if you need to continuously poll a directory for changes and trigger another task to run. This option should not be used when running in a chain.

To monitor a directory 24 hours a day, do the following:

Set the maximum wait time (**T1**) for the task = $24 \times 60 = 1440$ minutes. Schedule this task to run once a day (every 24 hours). i.e. when the previous schedule dies a new schedule is kicked off.

Using Directory Change Task in a Chain

- Create a "DirChange" task with the directory or file you need to monitor for changes.
- Create a Chain Task with the above "DirChange" task as the 1st step.
- Set the maximum wait time (**T2**) to be greater than **T1** above.
- Use the Exit Code criteria as (if Exit Code != -100 Go to Step 10) i.e. stop chain if exit code != -100
- Add a second task after this step.
- Set this Chain to run at time = X.
- Set another task type like (ftp, program etc.) to run at time = X+1 minutes
- On successful completion, the other task type (ftp, program etc.) should generate/copy/move/delete a file in the desired directory.
- If a directory change is detected, the DirChange task will exit with Exit Code = -100 and the Chain will run the 2nd task.
- If the task wait time (**T1**) is exceeded, the DirChange task will exit with Exit Code != -100 and the Chain will terminate without running the 2nd task.

Difference between Directory Change and Directory Monitor Tasks

Directory Change task keeps polling the directory for the maximum time set. When the maximum time expires, or a change is detected, this task either exits, or continues polling as a daemon. Can only monitor single directory, or a single file.

Directory Monitor task stores (in a log file), the modified date information for all files within the directory. This log file is read, and updated, with every run. The Directory Monitor task will return, after it compares the modified date information in the log file, with the actual modified dates of the files in the directory. Can also monitor sub directories.

Delete Files

The Delete task allows you to Delete file(s) within a single directory. You can also use the 'Local Monitor' task combined with the Delete task. This allows you to delete files based on file modified date or file size.

Directory

Enter the full path name of the directory where your files exists. Note that the path is case sensitive for unix Ftp servers.

Filename

Enter the name filter for the files you wish to copy [using the following wildcard rules](#). This field also supports [dynamic variables](#). Using dynamic variables, you can select a file based on current date/time.

Date Filter

If you need to filter files based on file modified date, select this option. For the between option, enter 2 values separated by '-' (2-4 etc.). Examples:

Older than 5 Minute, Newer than 2 Day, Between 3-5 Day, Between 1-4 Hour

Database Test

This is the most difficult task to implement correctly. Please read this entire documentation very carefully.

This utility helps you to test your connection to a database, on your system or, on any remote system. It connects to your database using an intermediate JDBC (Java Database Connectivity) driver. A database vendor or third party vendor provides this driver. **Detailed JDBC driver and connection information for popular databases like MSAccess, Oracle, Informix, MySql, mSql, Postgres are included in our knowledge base. The knowledge base is online at www.hiteksoftware.com.** Go to our main support page and use the knowledge base link. Our knowledge base will have information on these databases, and other databases provided by our users.

Detailed information on JDBC drivers, that are available for most databases, is provided by www.javasoft.com. Please read the JDBC driver documentation very carefully. The driver documentation will specify the values for the Driver Class Name and Database Url.

Database JDBC driver

For your database, you need to obtain the JDBC driver from your database vendor or other 3rd party vendor. The commercial database vendors have free JDBC drivers available at their web site. Legal issues may prevent us from redistributing some of these free JDBC drivers, in our software or, at our web site.

The JDBC driver is generally in the form of a .zip or .jar file. **You need to place the JDBC driver in the following folder depending on your installation method. Then restart the engine and the user interface.**

For installs including JVM : ...Automize/jre/lib/ext

For installs without JVM : ...jre/1.3/lib/ext or JDK1.3/jre/lib/ext etc.. depending on your system

Driver class name

Enter the class name for the driver. This information will be available, in the driver documentation. For Microsoft Access database, simply enter "access". Also, the JDBC driver required for MSAccess is included within java itself.

Database Url

Enter the Url connection information. This information will be available, in the Java JDBC driver documentation.

The Url format will vary with vendor, but generally includes the server name, port, and database name, username, and password.

Authorization

If your database requires authentication, then select this option and enter your username and password.

Username

Enter your username if required for database access. It is case sensitive.

Password

Enter your password if required for database access. It is case sensitive.

Directory Monitor

The Directory Monitor task is a special type of task which can be used in two ways:

- 1) If a change is detected in this task, you can run another task, or
- 2) The Directory Monitor task can also be embedded within a Chain task, and is useful for conditional task processing. If you want to use this task in a chain for conditional processing, select 'None' as the 'Task To Run'. If the criteria is satisfied, then an exit code = -100 is thrown. Else, an exit code = 0 is thrown.

The Directory Monitor task also outputs a variable called "TaskTitle::FileNames". This variable contains the list of all files that have been modified, since the last time this task was run. The format of this variable is #file1^#file2^#file3, where file1, file2, file3, are the names of the files, that have been modified. You can use this variable directly as the filename in other copy, ftp, or delete tasks, to process only these files.

Directory to Monitor

Enter the directory or file you need to monitor.

Include Subdirectories

To monitor all sub directories, select this option

Filename\$

Enter the name filter for the files you wish to copy [using the following wildcard rules](#). This field also supports [dynamic variables](#). Using dynamic variables, you can select a file based on current date/time.

Task To Run

Select the desired task to run if any of the selected criteria are met. If you want to use this task in a chain for conditional processing, select 'None'. You can also select multiple tasks to run in sequence, or to run simultaneously. To run tasks in sequence, use taskTitle1|taskTitle2|taskTitle3. To run tasks simultaneously, use taskTitle1&taskTitle2&taskTitle3.

Using Directory Monitor Task in a Chain

- Create a Directory Monitor task with the directory or file you need to monitor for changes.
- Create a Chain Task with the above "Directory Monitor" task as the 1st step.
- Use the Exit Code criteria as (Exit Code != -100 Go to Step 10) i.e. stop chain if exit code != -100
- Add a second task after this step.
- If a directory change is detected from comparison with the previous run of this task, the Directory

Monitor task will exit with Exit Code = -100 and the Chain will run the 2nd task.

- If no change is detected, the Directory Monitor task will exit with Exit Code = 0 and the Chain will terminate without running the 2nd task.

Difference between Directory Change and Directory Monitor Tasks

Directory Change task keeps polling the directory for the maximum time set. When the maximum time expires or a change is detected, this task either exits or continues polling as a daemon. Can only monitor single directory.

Directory Monitor task stores (in a log file), the modified date information for all files within the directory. This log file is read and updated with every run. **The Directory Monitor task will return as soon as it finishes reading, and comparing the modified date information in the log file with the actual modified dates of the files in the directory.** Can also monitor sub directories.

Email



The email task allows you to send email automatically to multiple recipients. You can also attach files.

Email Server Profile


Select the profile name for the email server. You should have previously created an email profile, using the Email Profiles menu item.

Adding a recipient to the current email message

You can add a recipient in 2 ways.

1. Enter the email address directly into the recipient text box. Hit the  button to add the recipient to the recipient list.
2. Select the recipient in the address book, then select the recipient type (To:, cc:, bcc:) and hit the paste button icon . This will add the recipient to the recipient list box.

Adding a recipient to the email address book

You can add a new recipient to the email address book, using the new button icon . Enter the desired name and email address. You have to hit enter, for any changes in the address book to be validated.

Subject \$

Enter the subject. This field also supports [dynamic variables](#). For example, you could automatically parse the current date, or computer IP address, into the subject.

Message Body \$

- 1) Enter the message body directly into the text field.
- 2) This field also supports [dynamic variables](#). For example, you could automatically parse the current date, or computer name, into the message.
- 3) Or, You can let the task dynamically parse the message text from a file on your system using the following format:
\$\$\$[message_file_path].
for example on windows, if the message file is c:\data\message.txt, use:
\$\$\$c:\data\message.txt

Adding Attachments \$

- 1) To add an attachment, type in the full path name of the file into the Attachment Text Box, or use the "Browse" button to select the file. The format for the attachments is: AbsoluteFilePath1,

AbsoluteFilePath2,.....

Example: c:\data\test.txt,c:\temp\data.htm

2) You can also specify multiple filenames in multiple directories, by using the following format:

[DIR]c:\your_dir[FILE]name_filter,.....

Enter the name_filter for the files you wish to transfer, [using the following wildcard rules](#)

Example: to attach all files that end with .txt from the c:\temp folder, use:

[DIR]c:\temp[FILE].txt

3) This field also supports [dynamic variables](#).

Example:

To attach all files in the c:\temp folder, which has the current date (example: 06-21-02) in their filename, use:

[DIR]c:\temp[FILE]\$%DATE::pp-dd-yy%\$

The date format in the filename has to match that entered above. See [dynamic variables](#) for more details

Append to Filename

You can append the date/time or a customized date to the attachments [based on current time/date, or custom code](#).

Headers

You can specify headers to add to the email. This will allow you to add mail priority, or other specific flags. An email server may not display all the headers you specify, or may not understand the headers you specify. For example, if you attempt to specify Email priority level, an email server may accept any, or all, or none of the following flags: X-Priority, X-MSMail-Priority, Importance, Priority.

To specify headers, use the following syntax:

header1=value1^header2=value2^header3=value3

Header Examples:

X-Priority=1^ Priority=Urgent

Importance=high

Reply-To=test@test.com

Email Logs

This administrative task will send you an email with the output log and activity log attached. You can save this task to the 'All Tasks' table, and then schedule this task. This is useful, if you deploy the software on a remote machine.

Email Server Profile

Select the profile name for the email server. You should have previously created an email profile, using the Email Profiles menu item.

Email Recipients

Enter the email address(es) to receive the email message. Separate multiple addresses by a comma ,.

File Monitor

The File Monitor task is a special type of task, which can be used in two ways:

- 1) If a change is detected in this task, you can run another task.
- 2) The File Monitor task can also be embedded within a Chain task, and is useful for conditional task processing. To use this task in a chain for conditional processing, select 'None' as the 'Task To Run'. If the criteria is satisfied, then an exit code = -100 is thrown. Else, an exit code = 0 is thrown.

The File Monitor task also outputs a variable called 'TaskTitle::FileNames'. This variable contains the list of all files that satisfy the monitoring criteria. The format of this variable is #file1^#file2^#file3, where file1, file2, file3 are the names of the files, that have been modified. You can use this variable directly as the filename, in other copy, ftp, delete tasks, to process only these files.

Directory

Enter the directory where your files are located

Filename\$

Enter the name filter for the files you wish to copy [using the following wildcard rules](#). This field also supports [dynamic variables](#). Using dynamic variables, you can select a file based on current date/time.

Task To Run

Select the desired task to run, if any of the selected criteria are met. If you want to use this task in a chain for conditional processing, select 'None'. You can also select multiple tasks, to run in sequence, or to run simultaneously. To run tasks in sequence, use taskTitle1|taskTitle2|taskTitle3. To run tasks simultaneously, use taskTitle1&taskTitle2&taskTitle3.

Options

The Task will be run, if any of the following selected criteria are satisfied. You can select 1 or more criteria.

- 1) File(s) which satisfy the Filename criteria exists
- 2) File(s) which satisfy the Filename criteria, as well as file size criteria
- 3) File(s) which satisfy the Filename criteria, as well as date modification criteria
- 4) Total files in directory are greater/less than certain value. This option does not update any of the dynamic variables.

Using File Monitor Task in a Chain

- Create a File Monitor task with the directory and file you need to monitor for changes.
- Create a Chain Task with the above "File Monitor" task as the 1st step.

- Use the Exit Code criteria as (Exit Code != -100 Go to Step 10) i.e. stop chain if exit code != -100
- Add a second task after this step.
- If a change is detected from comparison with the previous run of this task, the File Monitor task will exit with Exit Code = -100. The Chain will run the 2nd task.
- If no change is detected, the File Monitor task will exit with Exit Code = 0. The Chain will terminate without running the 2nd task.

Ftp

Before creating an Ftp task, we recommend that you use the FTP browser utility to get a connection. The ftp task allows you to transfer files from/to a server.

Ftp Profile

Select the ftp profile for the ftp server you need to connect to. You should have previously created an ftp profile, using the Ftp Profiles menu item.

Ftp Option

Use 'Get' to transfer files from the server, to your local system. Use 'Put' to transfer files from your local system, to the server. Use 'Delete' to delete files from the server.

Remote Directory

This is the most difficult parameter to enter correctly. We recommend that you use the FTP browser utility, to determine the value of the full path name, of the remote directory. Execute privilege is required for the remote directory, for all Ftp functions. Read privilege is required for Ftp Get. Write privilege is required for Ftp Put, and Ftp Delete functions. The path is case sensitive for unix Ftp servers. **If you are transferring files from the Default FTP login directory, you can leave this field blank.**

Local Directory

Enter the full path name of the local directory, where you need to put or get files.

Backup to Local Directory

Enter the full path name of the local directory, where you need to backup files. The Backup to local directory option does not support subdirectories. All files will be placed in the backup directory, without creating new subfolders in it.

Backup to Remote Directory

Enter the full path name of the remote directory, where you need to backup files. The Backup to remote directory option does not support subdirectories. All files will be placed in the backup directory, without creating new subfolders in it.

Use Staging

If you select this option, the file is first transferred to the local (get) or remote (put) folder entered. This folder acts as a staging folder. After the Ftp transfer is complete, the file is moved to the final destination folder. This option prevents processes from working on files, while the Ftp transfer is in progress. In the staging field, you should enter the full path name, of the final destination directory. The staging option does not support subdirectories. All files will be placed in the final destination directory, without creating new subfolders in it.

Filename\$

Enter the name filter for the files you wish to copy [using the following wildcard rules](#). This field also supports [dynamic variables](#). Using dynamic variables, you can select a file based on current date/time.

Transfer Type

File transfer type can be either Ascii or Binary. Binary mode will transfer .exe, .jpg or other binary files. Binary mode should be the default transfer mode. Ascii mode can be used to transfer text or html files. If you transfer binary files in ascii mode, you will not get the entire file.

Append date/time To filename

You can choose to [append either the date, or the date/time, or custom code](#). If your filename has an extension, the date or date and time is appended before the extension. This feature is useful for archive purposes.

Maintain Timestamp (for Get only)

The task can reset the transferred local files timestamp, to match the remote files timestamp. This applies ONLY to the GET option.

Reset Local Timestamp (for Put only)

There is no Ftp standard, to reset the remote files timestamp, to match the local files timestamp after an Ftp PUT. Hence, we provide the option to reset the source local files timestamp, to match the newly transferred remote files timestamp. There is a new proposal, to reset the remote file timestamp, using the MFMT (Modify file modification time) command. However, not many Ftp servers support the MFMT command. (Many Ftp servers incorrectly allow setting the remote file timestamp, using the MDTM command. This MDTM command is not an Ftp standard to set the remote file timestamp. The Ftp task does not reset the timestamp through this MDTM command.)

Delete Source file

You can choose to delete the source file, after a successful transfer. If the ftp transfer fails, the source file will not be deleted.

Transfer with temporary extension

The file is transferred with a temporary extension, and then renamed to the desired name, after the transfer is complete. This is useful, if you have automated scripts/programs, which continuously work on transferred files, with certain extensions. This ensures that, the automated script will not work on a partially transferred file.

Log Transfer

You can choose to log the transfer into the [Ftp Log](#).

Transfer Modified files

If you choose this option, files are transferred, only if the source file, is newer than the target file. You can specify an offset to be added to the remote ftp files timestamp, while comparing the timestamps.

Ftp server time offset

Sometimes, a Ftp server is in a different timezone from your local system. You can specify an offset to be considered while determining which file (local or remote) is newer. The remote file timestamp is not changed. Enter a positive number to add an offset to the actual remote ftp file timestamp. Enter a negative number to subtract an offset from the actual remote ftp file timestamp. For example, consider a file with:

Actual Local file timestamp = June 1, 2005, 10:00 PM

Actual Remote file timestamp = Jun 1, 2005, 10:30 PM

Offset = 0

Put Option: File will not be transferred, since remote file is newer than local file

Get Option: File will be transferred, since remote file is newer than local file

Offset = -60 (Remote file timestamp is now calculated as 9:30 PM)

Put Option: File will be transferred, since calculated remote file timestamp is older than local file timestamp.

Get Option: File will not be transferred, since calculated remote file timestamp is older than local file timestamp.

Transfer Modified files using Ftp Log

If you choose this option, files are transferred, only if the source file date, is newer than the source file date entry in the Ftp Log. For this option to work correctly, you should also enable the 'Log Transfer' option. If the source file path is not found in the Ftp log, then the file is always transferred.

Date Filter

If you need to filter files based on file modified date, select this option. For the between option, enter 2 values separated by '-' (2-4 etc..). Examples:

Older than 5 Minute, Newer than 2 Day, Between 3-5 Day, Between 1-4 Hour

File Permissions

For the PUT option, you can reset the permissions of the file, after the file is transferred to the server. The site must support the "SITE chmod" command. This is normally valid only on UNIX-based systems.

Examples for field entry:

644 = rw- for USER, r-- for GROUP, r-- for WORLD

755 = rwx for USER, r-x for GROUP, r-x for WORLD

754 = rwx for USER, r-x for GROUP, r-- for WORLD

where r = read, w = write, and x = execute permissions.

Ftp Commands

This task allows you to connect to an ftp server and issue commands. There are two types of commands supported:

- 1) The standard ftp commands as defined in RFC 959. [Please see list of ftp commands.](#) Only the most useful commands are listed here. Please search the internet for RFC 959 if you need more information on all the standard Ftp commands.
- 2) Custom functions provided by the Ftp Command task. [Please see list of custom functions.](#)

Please note that the ftp commands, that you normally use in the ftp commandline tools, in MSDOS, MacOSX or Unix etc. are not necessarily supported. For example, the LS, CD, Get or Put commands, in many commandline ftp tools are not valid RFC 959 ftp commands. Get and Put, are commands created by many commandline ftp tools, to implement the RETR and STOR commands, of RFC 959. To put or get files, you can use the custom functions \$\$GET& \$\$PUT that are provided by this task.

This task is multithreaded. So you can run multiple Ftp Command tasks simultaneously.

This task also outputs dynamic variables for each response provided by the Ftp server. The variables are of the form TaskTitle::ResponseX, where X is the response from the server, for each of your Ftp commands. You can use these response variables in a Variable Monitor task to do conditional processing.

Host

Enter the host name or IP address of the Ftp server. If you have to connect through an Ftp proxy server, or firewall, enter the host name, or IP address of the firewall, or proxy server.

Port

Enter the desired port. Default Ftp port = 21

Password

In the password field, enter your password required to connect. Then in the command list, use the following syntax to send the hidden password: \$\$LOGIN::username::\$PASSWORD\$, or if you need to manually login, use: PASS \$PASSWORD\$. The string \$PASSWORD\$ will be replaced by the value in the password field. If you have no reason to hide your password, you do not need to fill in the password field.

Commands

Enter your ftp commands, 1 per line. The custom commands are case sensitive. First line should normally be the USER command, and second line should normally be the PASS command. This field also supports [dynamic variables](#). Using dynamic variables, you can enter a command, which will be dynamically resolved at runtime. example: cd /users/\$%USERNAME%\$

Example1

USER tom sends username
 PASS \$PASSWORD\$ sends password
 ACCT production sends account information
 PWD prints current remote ftp directory
 CWD /users/tom sets /users/tom as the current ftp directory
 \$\$LCD::c:\test sets c:\test as the current local directory
 \$\$MPUT::test puts all files from c:\test, which include 'test' in their name, to /users/tom
 \$\$MGET:: gets all files from /users/tom to c:\test

Example2

\$\$LOGIN::user::\$PASSWORD\$ logon information
 PWD prints current remote ftp directory
 CWD /users/tom sets /users/tom as the current ftp directory
 \$\$LCD::c:\test sets c:\test as the current local directory
 \$\$PASV sets the passive mode of transfer
 \$\$GET::test.txt gets test.txt from /users/tom to c:\test
 \$\$PUT::test.txt puts test.txt from c:\test to /users/tom

List of custom commands

<p>\$ \$\$LOGIN</p>	<p>Description: Simple logon using username and password information Syntax: \$\$LOGIN::username::password Examples: \$\$LOGIN::jim::jim04 or \$\$LOGIN::jim::\$PASSWORD\$ Comments: To hide password, use \$PASSWORD\$ and enter password in password field For more complicated logons or to connect through firewalls, you may need to use as required: USER, PASS, ACCT, SITE, OPEN etc..</p>
<p>\$\$LCD</p>	<p>Description: Sets the current local folder Syntax: \$\$LCD::path_to_local_folder Example: \$\$LCD::c:\data\test Comments: Very important. Should always be set before transferring files Comments: Internal module use only. Ftp server has no idea about current local folder Comments: Used internally by \$\$PUT, \$\$GET, \$\$MPUT, \$\$MGET, \$\$MDEL functions</p>
<p>\$\$PASV</p>	<p>Description: Sets the passive mode of transfer. Syntax: \$\$PASV</p>
<p>\$\$PUT</p>	<p>Description: Puts a single file from current local folder, to current remote ftp folder Syntax: \$\$PUT::filename Example: \$\$PUT::test.txt Comments: Only single file is put, file should exist in current local folder</p>
<p>\$\$GET</p>	<p>Description: Gets a single file from current remote ftp folder, to current local folder Syntax: \$\$GET::filename Example: \$\$GET::test.txt Comments: Only single file is got, file should exist in current remote ftp folder</p>

\$\$PPUT	<p>Description: Puts a single file from local path, to ftp path</p> <p>Syntax: \$\$PPUT::path_to_local_file::path_to_remote_file</p> <p>Example: \$\$PPUT::c:\data\test.txt::/users/tom/test5.txt</p> <p>Comments: Only single file is put, file path should exist in the local system</p>
\$\$PGET	<p>Description: Gets a single file from ftp path, to local path</p> <p>Syntax: \$\$PGET::path_to_remote_file::path_to_local_file</p> <p>Example: \$\$PGET::/users/tom/test5.txt::c:\data\test.txt</p> <p>Comments: Only single file is got, file path should exist in the remote system</p>
\$\$MPUT	<p>Description: Puts multiple files from current local folder, to current ftp folder</p> <p>Syntax: \$\$MPUT::filename_filter</p> <p>Example: \$\$MPUT:: (will put all files from current local folder)</p> <p>Example: \$\$MPUT::test (will put all files which include test in their name)</p> <p>Example: \$\$MPUT::.txt^.htm (will put all files which have .txt or .htm extensions)</p> <p>Comments: Enter the filename_filter using the following wildcard rules</p>
\$\$MGET	<p>Description: Gets multiple files from current ftp folder, to current local folder</p> <p>Syntax: \$\$MGET::filename_filter</p> <p>Example: \$\$MGET::* (will get all files from current ftp folder)</p> <p>Example: \$\$MGET::test (will put all files which include test in their name)</p> <p>Example: \$\$MGET::.txt^.htm (will put all files which have .txt or .htm extensions)</p> <p>Comments: Enter the filename_filter using the following wildcard rules</p> <p>Comments: Automatically detects directory listing style and get directory information</p>
\$ \$MSGET	<p>Description: Gets multiple files from current ftp folder to current local folder</p> <p>Syntax: \$\$MSGET::* (will get all files from current ftp folder)</p> <p>Example: \$\$MSGET::test (will put all files which include test in their name)</p> <p>Example: \$\$MSGET::.txt^.htm (will put all files which have .txt or .htm extensions)</p> <p>Comments: Enter the filename_filter using the following wildcard rules</p> <p>Comments: Only obtains names of objects in current ftp directory This function does not try to detect if object is a File, or a Directory. It will also try to GET directories, if they satisfy the filename filter. This results in the following error being output, for every Directory in current ftp folder: 550 dir_name: [Error information varies by ftp server]. This is expected behaviour.</p>
\$\$MDEL	<p>Description: Deletes multiple files from current ftp folder</p> <p>Syntax: \$\$MDEL::filename_filter</p> <p>Example: \$\$MDEL:: (will delete all files from current local folder)</p> <p>Example: \$\$MDEL::test (will delete all files which include test in their name)</p> <p>Example: \$\$MDEL::.txt^.htm (will delete all files which have .txt or .htm extensions)</p> <p>Comments: Enter the filename_filter using the following wildcard rules</p>

List of useful ftp commands - RFC 959

HELP	<p>Description: Returns list of commands or details for a single command</p> <p>Syntax: HELP [command]</p> <p>Example: HELP returns a list of commands supported by the ftp server</p> <p>Example: HELP USER returns details on the USER command</p>
USER	<p>Description: Sends the username to begin the login process</p> <p>Syntax: USER userID</p> <p>Example: USER tom</p>
PASS	<p>Description: After sending the USER command,send this command to complete login</p> <p>Syntax: PASS user_password</p> <p>Example: PASS ergts56r</p>
PASV	<p>Description: Sets the passive mode of transfer.</p> <p>Syntax: PASV</p>
TYPE	<p>Description: Specifies ascii or binary mode of data transfer</p> <p>Syntax: TYPE mode</p> <p>Example: TYPE I or TYPE A</p> <p>Comments: A - ASCII text, E - EBCDIC text, I - image (binary data)</p>
PORT	<p>Description: Specifies host & port to which the ftp server should connect for next data transfer</p> <p>Syntax: PORT h1,h2,h3,h4,p1,p2</p> <p>Example: PORT 192,168,0,121,15,196</p> <p>Comments: ip address = h1.h2.h3.h4, port = p1*256 + p2</p>
ACCT	<p>Description:Sends the account required for login</p> <p>Syntax: ACCT account_name</p> <p>Comments: normally sent after the PASS command. Uncommon command</p>
CWD	<p>Description: Sets the current ftp directory on remote ftp server</p> <p>Syntax: CWD remote_ftp_path</p> <p>Example: CWD /users/tom/data</p>
CDUP	<p>Description: Sets the parent of the current ftp folder to be the current ftp folder</p> <p>Syntax: CDUP</p>
DELE	<p>Description: Deletes a single file from the current ftp folder</p> <p>Syntax: DELE filename</p> <p>Example: DELE test.txt</p>
RNFR	<p>Description: Used to rename a file. Specifies the file to be renamed</p> <p>Syntax: RNFR original_name</p> <p>Example: RNFR test.txt</p> <p>Comments: Should be followed by RNTO command</p>
RNTO	<p>Description: Used to rename a file. Specifies the new name for the file</p> <p>Syntax: RNTO new_name</p> <p>Example: RNTO new.txt</p> <p>Comments: Should be preceded by RNFR command</p>

SITE	Description: Used to issue a site specific command Syntax: SITE command Example: Use 'HELP SITE' to see list and syntax of supported commands Comments: May not be supported by many Ftp servers
RETR	Description: Transfers file from remote host Syntax: RETR filename Example: RETR test.txt Comments: Should be preceded by the PORT or PASV commands
STOR	Description: Transfers file to the remote host Syntax: STOR filename Example: STOR test.txt Comments: Should be preceded by the PORT or PASV commands

Ftp Monitor

You can monitor a remote ftp folder for changes, or other criteria. This task can be used in two ways:

- 1) If a change is detected in this task, you can run another task, or
- 2) The Ftp Monitor task can also be embedded within a Chain task, and is useful for conditional task processing. If you want to use this task in a chain, for conditional processing, select 'None' as the 'Task To Run'. If the criteria is satisfied, then an exit code = -100 is thrown. Else, an exit code = 0 is thrown.

The Ftp Monitor task also outputs a variable called 'TaskTitle::FileNames'. This variable contains the list of all files that satisfy the monitoring criteria. The format of this variable is #file1^#file2^#file3, where file1,file2,file3 are the names of the files, that have been modified. You can use this variable directly, as the filename in other copy, ftp, delete tasks to copy/ftp/delete only these files. This variable is only updated if the filename, filesize, or filedate criteria options are selected.

Directory

Enter the remote folder that you wish to monitor

Filename\$

Enter the name filter for the files you wish to copy [using the following wildcard rules](#). This field also supports [dynamic variables](#). Using dynamic variables, you can select a file based on current date/time.

Task To Run

Select the desired task to run, if any of the selected criteria are met. If you want to use this task in a chain for conditional processing, select 'None'. You can also select multiple tasks, to run in sequence, or simultaneously. To run tasks in sequence, use taskTitle1|taskTitle2|taskTitle3. To run tasks simultaneously, use taskTitle1&taskTitle2&taskTitle3.

Options

The Task will be run, if any of the following selected criteria are satisfied. You can select 1 or more criteria.

- 1) Any change is found in the directory, since the last run of this task. This option does not update any of the dynamic variables.
- 2) File(s) which satisfy the Filename criteria exists
- 3) File(s) which satisfy the Filename criteria, as well as file size criteria
- 4) File(s) which satisfy the Filename criteria, as well as date modification criteria
- 5) Total files in directory are greater/less than certain value. This option does not update any of the dynamic variables.

Ftp Server Address

The Ftp server address, is the name, or the IP address, of the Server on the network. Please check this value, from your network administrator, or Internet service provider, or Ftp server administrator.

Server Port

The default ftp command port is 21.

Username and Password

If the Ftp server is private, you will typically need a username/password to transfer files. Please check with the Ftp server administrator, for your user account information.

Directory Listing Style

Ftp servers provide directory listing in various formats. The most common is the Unix style. We currently support Unix and DOS listing styles. If your Ftp server provides a listing in some other style, the task may fail. However, most Ftp servers have an option to provide directory listing in the Unix style. Please request your Ftp administrator, to set up your Ftp account, to receive listing in Unix style if possible.

If your Ftp server does not provide a listing in the Unix or DOS style, you can try the NameOnly option. Here, only a list of names in the directory is obtained. There is no information on the size, date, or if the name corresponds to a file, or a directory. Hence, you cannot transfer files based on date, or size. Also, since the file or directory information is not available, subdirectory (tree) transfers are not possible in this mode.

Passive Mode

In normal mode, the Ftp server creates the data connection. In Passive mode, your local system will create the data connection. You may need to use passive mode, if you are behind a firewall/proxy server, or you get erratic transfers, or failed data channel errors.

Using Ftp Monitor Task in a Chain

- Create a Ftp Monitor task with the remote directory you need to monitor for changes.
- Create a Chain Task with the above "Ftp Monitor" task as the 1st step.
- Use the Exit Code criteria as (Exit Code != -100 Go to Step 10) i.e. stop chain if exit code != -100
- Add a second task after this step.
- If any of the selected criteria are met, the Ftp Monitor task will exit with Exit Code = -100 and the Chain will run the 2nd task.
- If no change is detected, the Ftp Monitor task will exit with Exit Code = 0 and the Chain will terminate without running the 2nd task.

Filing Variables

This feature allows you to output the value of a variable, to a local file on your system. You can then use this file in other tasks, or your own programs. You can also insert the variable into various locations of a file, depending on certain criteria you select. This feature can be useful in many situations, examples:

- 1) monitoring your own log files for 'errors' which occur today or within the last hour etc.
- 2) appending one file to another.

Variable name

Select the variable you need to monitor from the list. For some variables like DATE, you need to manually enter the format. In version 6.x, you need to enter the variable name surrounded by `%%...`. In version 5.x, the surrounding `%%...` was not required. By adding the `%%...` into the format, it is now possible to set the Variable Name field as an entire line of text which includes the variable.

Examples of Variable Name Field entries:

Today's date is `%%DATE::qqq-dd_hh:mm%%`

my username is `%%USERNAME%%`

This ip address = `%%IPADDRESS%%`

Filepath

Enter the full file path of the file, which you want to save the variable to. This field supports dynamic variables.

Filing Options

Choose your filing criteria. You can insert the value of the variable into various locations. If you choose the 'Line Number' or one of the 'String containing' options, you should enter the correct 'Option Value' also.

Option Value

This Option Value depends on the Filing Option above.

Examples:

Variable Name = `%%DATE::qqq-dd_hh:mm%%`

Filing Option = Insert at Line Number

Option Value = 10

This will insert the date/time at line 10 in the file

or

Variable Name = `%%DATE::qqq-dd_hh:mm%%`

Filing Option = Insert before first line containing

Option Value = error

This will insert the date/time in the file, before the first line that contains the word 'error'

Types of variables listed:

Task Variables: Task variables values are generated/updated when the tasks run. Example: Tasks exit codes are stored as the following variable: TaskTitle::ExitCode

System Variables: System variables which are resolved when the current task runs, examples are DATE, USERNAME etc..

Java Variables: Java variables that are available to the Java Virtual Machine that the program runs in. You can get the value of java variables from the Utilities menu / Java Variables

User Variables: Users can add their own variables via batch files, scripts or programs. This way users can link output from their own programs to this program.

Log File Monitor

The Log file monitor task allows you to monitor a log file for certain alert terms. If the alert term is found, then the timestamp for the log entry is determined and archived. If the timestamp is newer than the previously archived timestamp, you can choose to run another task(s). Also, an exitcode = -100 is thrown when a new alert is found.

FilePath

Enter the full path to the log file, that needs to be monitored.

Alert Terms \$

Enter the alert term required. If you have more than 1 alert term, separate them by a character, not included within the alert term. Example: Searching^Java^Automation . This field also supports [dynamic variables](#). Using dynamic variables, for example, you can search a logfile for today's date/time, and run a task if that entry is found.

Alert Term Separator

Enter the separator character used between alert terms, if you have more than 1 alert term.

Logging Direction

New logging entries can either be inserted at the top, or added at the bottom.

Task To Run

Select the desired task to run, if any of the selected criteria are met. If you want to use this task in a chain, for conditional processing, select 'None'. You can select multiple tasks to run in sequence or simultaneously. To run tasks in sequence, use taskTitle1|taskTitle2|taskTitle3. To run tasks simultaneously, use taskTitle1&taskTitle2&taskTitle3.

Logging Date Format

Enter logging date format, using the following rules below. Please read this entire section very carefully, and please read all the Examples, at the bottom of this section. If you make any mistake in entering the format, the task will not be able to read the timestamp and the task will fail.

Rules:

- 1) The log file should have a standardized log format. The timestamp should be in the same column location, for all lines, else the timestamp in that particular line, will not be parsed.
- 2) The format is: variable1=xxxx(-)variable2=yyyyyy(-)...etc.
- 3) One of these types of formats can be specified a) columnformat or b) tokenformat

formattype = columnformat/tokenformat

For tokenformat, use the following variables

formattype = tokenformat

tokens = separators that are used to separate fields. Multiple separators can be specified, one after another. Example: tokens=SPACE*&-_(-) would use the following characters as separators: whitespace * & - _

tokcol_x = specify the format for tokcol_x. Please see examples at bottom of this page for details.

For columnformat, use the following variables

formattype = columnformat

colsep = separator between columns in the log file. Enter SPACE if the column separator is white-space or Enter TAB if the columns are separated by tabs. Example: colsep=SPACE(-) or colsep=,(-)

datecol = Column where date is logged. Example: datecol=4(-)

timecol = Column where time is logged. Example: timecol=4(-)

dateformat = date format used. Example: dateformat=YY-qq-dd(-)

timeformat = time format used. If datecol is same as timecol, leave this field empty. Example: timeformat= hh:mm:ss(-)

tokens = separators between fields. use all separators between date components and time components as well as the separator between date and time, if datecol is same as timecol

Examples:

if date is like 03-15-2004, and time is like 09:15:05, tokens=-:(-)

if date and time are logged into same column like, 03-15-2004_09:15:05, tokens=_-:(-)

Specifying date/time formats

Use the same rules, as that used for the DATE variable. Example:

If date time logging output = Feb 3, 2004 4:44:07 PM

Date time format required = Qqq ddx, YY hhx:mm:ss am_pm

YY = 4 digit year is always output (2001 etc..)
yy = 2 digit year is always output (00-99)
pp = 2 digit month is always output (01-12)
ppx = 1 or 2 digit month can be output(1-12)
qqq = 3 character month is always output (jan-dec)
QQQ = 3 character month is always output (JAN-DEC)
Qqq = 3 character month is always output (Jan-Dec)
dd = 2 digit date is always output (01-31)
ddx = 1 or 2 digit date can be output(1-31)
HH = 2 digit hour based on 12 hour clock is always output (01-12) (should generally be used with am_pm)
HHx = 1 or 2 digit hour based on 12 hour clock can be output(1-12)
hh = 2 digit hour based on 24 hour clock is always output (00-23)
hhx = 1 or 2 digit hour based on 24 hour clock can be output (0-23)
mm = 2 digit minute is always output (00-59)
mmx = 1 or 2 digit minute can be output (0-59)
ss = 2 digit seconds is always output (00-59)
ssx = 1 or 2 digit seconds can be output (0-59)
am_pm = AM is output for AM hours, PM is output for PM hours

Examples of Logging date format field entries

1) NCSA Combined / NCSA Extended / Extended Log Format / Microsoft Extended Format.

Example log entry:

2667 avi.start.com - Authorize 401 /pi.admin 1998-03-09 15:21:37 2345

ColumnFormat Logging date format field should be:

formattype=columnformat(-)colsep=TAB(-)tokens=-:(-)datecol=7(-)timecol=8(-)dateformat=YY-dd-qq
(-)timeformat=hh:mm:ss(-)

TokenFormat Logging date format field should be:

formattype=tokenformat(-)tokens=-:TAB(-)tokcol_7=YY(-)tokcol_8=pp(-)tokcol_9=dd(-)tokcol_10=hh
(-)tokcol_11=mm(-)tokcol_12=ss(-)

2) NCSA - Common Log File format:

Example log entry:

255.255.255.255 - REDMOND\doug [07/Jun/1996:17:39:04 -0800] "POST /iisadmin/default.htm?-,
HTTP/1.0" 200 3401

ColumnFormat Logging date format field should be:

formattype=columnformat(-)colsep=space[(-)tokens=:/(-)dateformat=dd/Qqq/YY:dd:mm:ss(-)

timeformat=(-)datecol=4(-)timecol=4(-)

TokenFormat Logging date format field should be:

formattype=tokenformat(-)tokens=[/:SPACE (-)tokcol_5=dd(-)tokcol_6=Qqq(-)tokcol_7=YY(-)
tokcol_8=hh(-)tokcol_9=mm(-)tokcol_10=ss(-)

3) Microsoft IIS Log Format

Example log entry:

255.255.255.255, user_name, 03/20/98, 23:58:11, MSFTPSVC, SALES1, 255.255.255.255, 60, 275, 0,
0, 0, PASS, intro.htm

ColumnFormat Logging date format field should be:

formattype=columnformat(-)colsep=,SPACE(-)tokens=:/(-)dateformat=dd/Qqq/YY(-)timeformat=dd:
mm:ss(-)datecol=3(-)timecol=4(-)

TokenFormat Logging date format field should be:

formattype=tokenformat(-)tokens=,SPACE/:(-)tokcol_3=pp(-)tokcol_4=dd(-)tokcol_5=YY(-)
tokcol_6=hh(-)tokcol_7=mm(-)tokcol_8=ss(-)

4) WebSTAR Log format

Example log entry:

03/09/98 15:21:37 ok avi.start.com :pi.admin 2667

ColumnFormat Logging date format field should be:

formattype=columnformat(-)colsep=TAB(-)tokens=:/(-)dateformat=dd/pp/yy(-)timeformat=hh:mm:ss(-)
datecol=1(-)timecol=2(-)

TokenFormat Logging date format field should be:

formattype=tokenformat(-)tokens=TAB/:(-)tokcol_1=pp(-)tokcol_2=dd(-)tokcol_3=yy(-)tokcol_4=hh(-)
tokcol_5=mm(-)tokcol_6=ss(-)

5) No tokens format, date and time in same column

Example log entry:

03292004-152137 Test task - exit=0

Only ColumnFormat Logging can be used as follows:

formattype=columnformat(-)colsep=SPACE(-)tokens=(-)dateformat=ppddYY-hhmmss(-)timeformat=(-)
datecol=1(-)timecol=1(-)

tokens should always be blank, if either the date or time components, are not separated by a separator

character, i.e. tokens=(-)

6) No tokens format, date and time in different columns

Example log entry:

03292004 152137 Test task - exit=0

Only ColumnFormat Logging can be used as follows:

formattype=columnformat(-)colsep=SPACE(-)tokens=(-)dateformat=ppddYY(-)timeformat=hhmmss (-)

datecol=1(-)timecol=2(-)

tokens should always be blank, if either the date or time components, are not separated by a separator

character, i.e. tokens=(-)

Maintenance

This feature does the following:

- 1) It requests the Java Virtual Machine to do memory cleanup.
- 2) It saves user, task and schedule data.
- 3) Re-launches the Scheduler Engine automatically.

You can create and schedule a Maintenance task weekly/monthly to free memory, if you notice that the memory consumption by the Engine is high. ***Please create only a single Maintenance task. Make sure that no other task is scheduled, when this task has a scheduled run. Leave at least 1 minute idle time, before this task runs.*** Generally, you may need to run this task only on Windows and Mac OS. This Maintenance task is also useful, if you are running tasks very frequently (every few seconds), and running out of memory. In this case, you can schedule a Maintenance task more frequently, depending on your needs.

On Solaris and Linux, we have run tasks every 10 seconds for 1 month continuously, without running out of memory, or the engine shutting down. The memory usage, during this accelerated reliability testing, stabilized at a peak of ~50 MB, for the entire 1 month testing period. Normally, the memory usage should cycle up and down between 15-30 MB.

About Memory Usage

This program does not handle any of the low-level memory allocation and release. Memory allocation and release is done by Java. Java prevents the programmer, from accessing all low-level memory handling routines, making memory safe. This prevents memory leaks and corruption, within the program, or in other running applications. This prevents serious problems, that can occur with C or C++ programs, like memory access violations, page faults, illegal operations, system crashes etc. Java has automatic memory recycling, called garbage collection. Hence, you should see the memory usage, cycle up and down between 15-30 MB. However, Java tends to consume lots of memory, and may not release it efficiently on some Windows and Mac OS systems. The default maximum memory allowable is 64 MB. Sometimes, Java will keep requesting more memory, from the operating system. Java may shut down automatically, when the maximum allowable memory (64MB) is exceeded. This is due to incomplete garbage collection, by the Java machine, on certain operating systems.

Ping



This task allows you to check, if a server is open to connection, on the specified port. Each ping result is logged by the task. The uptime statistics, for the server:port combination, can be viewed via the Logs / TaskLogs window. The statistics filename is "TaskTitle_statistics".

The "Ping" protocol is not actually implemented. Only a simple socket connection, to the specified host is attempted. **For safety purposes, you can only ping a server every 1 minutes.** If the server is pinged twice in less than a minute, the socket connection will not be attempted, and an Exit Code = 5 is returned.

Host

Enter the host name or IP address.

Port

Enter the desired port. Common ports are: SMTP=25, FTP=21, HTTP=80, TELNET=23, POP3=110.

Ping Statistics

When a ping task finishes running; a special log file named "TaskTitle_statistics" is generated. You can view this statistics file, by clicking on the filename, in the Logs / TaskLogs Window. This allows you to maintain uptime statistics of your server.

Resetting the Ping Statistics File

Go to the Logs menu, and select the "Manage Logs" menu item. Select the statistics file for your ping task, and hit delete. The ping statistics will be cleared, and will start afresh, the next time the task is run.

Print Files

The Print task allows you to print text files only.

Directory

Enter the full path name of the directory, where your files exists. Note that the path is case sensitive for unix Ftp servers.

Filename\$

Enter the name filter for the files you wish to copy [using the following wildcard rules](#). This field also supports [dynamic variables](#). Using dynamic variables, you can select a file based on current date/time.

Synchronize Folders

This task allows you to synchronize two folders.

Action

Three synchronization modes are supported. Be careful when using the Unidirectional Mirror mode, because newer files can be replaced by older files, and unpaired files are deleted.

1) Bi-directional Mirror

All unpaired files and folders in the Source are copied to the Target.

All unpaired files and folders in the Target are copied to the Source.

Older files in the Target are replaced with their newer counterpart in the Source.

Older files in the Source are replaced with their newer counterpart in the Target.

2) Unidirectional Copy

All unpaired files and folders in the Source are copied to the Target.

Older files in the Target are replaced with their newer counterpart in the Source.

3) Unidirectional Mirror

All unpaired files and folders in the Source are copied to the Target.

Older files in the Target are replaced with their newer counterpart in the Source.

Newer files in the Target are replaced with their older counterpart in the Source.

All unpaired files and folders in the Target are removed.

Source Directory

Enter the full path name of the directory, where your source files exist. Note that the path is case sensitive for unix Ftp servers.

Target Directory

Enter the full path name of the directory, where you need to put the target files. Note that the path is case sensitive on unix systems.

Include subdirectories

If you select this option, all subdirectories and files within the Source Directory are copied to the Target Directory. If a subdirectory does not exist within the target folder, then it is automatically created.

Filename \$

Leave this field blank if you want all files to be synchronized. Enter the name filter for the files you wish

to synchronize, [using the following wildcard rules](#) . This field also supports [dynamic variables](#). Using dynamic variables, you can select a file based on current date/time.

Folder Filter\$

Leave this field blank if you want all folders to be synchronized. If you need specific folders to be synchronized, then enter the folder name, [using the following wildcard rules](#). If a folder passes the filter, then all its subfolders will also pass the filter.

Text Search

This task allows you to search for a string of text (or multiple strings), within a file (or within multiple files), in a directory. If the text is found, then the line containing the text, is appended to a results file. You can also choose to append surrounding lines, to the results file.

You can use this task, to search for text in web pages, or in files located on a ftp server. First run the web download, or ftp download (get) task, and download the file(s) to a local directory. Then run this Text Search task on the downloaded file(s).

Source Directory

Enter the full path name of the directory, where the file(s) you need to search are located.

Filename\$

Enter the name filter for the files you wish to copy [using the following wildcard rules](#). This field also supports [dynamic variables](#). Using dynamic variables, you can select a file based on current date/time.

Search Term\$

Enter the search term required. If you have more than 1 search term, separate them by a character, which is not included within the search term. Example: Searching^Java^Automation . This field also supports [dynamic variables](#). Using dynamic variables, for example, you can search a logfile for toady's date/time, and run a task if that entry is found.

Search Term Separator

Enter the separator character used between search terms, if you have more than 1 search term.

Surrounding Lines

If a search term is found, the line is appended to the results file. If you need to append surrounding lines also, select the desired number of surrounding lines to append.

Results Directory

Enter the full path name of the directory, where you need to put the results File. Note that the path is case sensitive on unix systems.

Results File\$

Enter the name for the results File. If left blank, a default filename of your task title is used. This field also supports [dynamic variables](#). Using dynamic variables, you can save the file to a customized name based on date/time.

Append to Filename

You can save each archive to a different filename [based on current time/date, or based on custom code you set.](#)

Telnet

This task allows you to telnet to a server and issue commands. If the server is up an exit code = 0. If server is down or unreachable, exit code = 1. No other exit codes are thrown for login failures etc.

Host

Enter the host name or IP address.

Port

Enter the desired port. Default Telnet port =23.

Delay

Enter the delay time between sending your commands. 1 second is minimum. 5 seconds is good enough, unless you are on a slow network. You can increase it to ~15, but make sure you do not exceed the hosts telnet time out value.

Command\$

Enter your telnet commands, 1 per line. First line should be your username, Second line should be your password and the last line should be **exit**. This field also supports [dynamic variables](#). Using dynamic variables, you can enter a command, which will be dynamically resolved at runtime. example: cd /users/ \$%USERNAME%\$

Example of Commands to enter

```
myUsername  
myPassword  
ls -lt  
exit
```

Telnet(Adv)

This allows you to telnet to a server and issue commands. This Telnet(Adv) task is more advanced than the Telnet task. It will try and detect the presence of the telnet login prompts, and the telnet command prompts, before sending the next command. If these prompts are not detected, within the timeout value set, it will send the next command

Host

Enter the host name or IP address.

Port

Enter the desired port. Default Telnet port =23.

Username

Enter your username.

Password

Enter your password.

Polling Period

Enter the desired polling time. This polling period applies only to Telnet commands, but not to the Login commands. Polling period does not apply to username and password commands. During the Login commands, the polling period is fixed at 1 second. For the regular Telnet commands, the module will poll the connection, at the Polling interval, for the reply string from the server. If the last 10 characters (default value) of the reply string from the server contain the Telnet prompt, then the next command is sent. This polling period should be less than the command time.

Login Prompt

This is usually the last character, that the server outputs, when it requests a username or password to be sent. This value is normally : but may depend on the telnet server. The task will look for this character, in the last 10 characters (default value) of the reply string from the server. If this character is found, the next command is sent. If the login prompt character is not found, within the Login time set, the next command is sent.

Telnet Prompt

This is usually the last character, that the server outputs, when it is done processing the current telnet command. This value normally depends on the system shell you are connecting to. The task will look for telnet prompt character, in the last 10 characters (default value) of the reply string from the server. As soon as this character(s) is found, the next command is sent. If the telnet prompt character is not found, within the telnet command time set, the next command is sent.

Prompt Characters

This module waits for the Login or Telnet prompts, before sending the next command. It will search the last reply string from the telnet server, for the Login prompt, or the Telnet prompt. You can define how the module should search for these prompts. If you select Prompt Characters = 10 (default value), then the module will search the last 10 characters, of the last reply string. If the Login prompt, or the Telnet prompt, is found in these 10 characters, then the next command is sent. If your server sends a bunch of characters, after the Telnet prompts (Example: MS Telnet server), then you may have to increase this value to 1000, to detect the Telnet prompt. As soon as this character(s) is found, the next command is sent.

Login Time

Enter the maximum time allowed for each login command (username and password). If this login time has been exceeded, and the server has not replied back with a telnet login prompt, then the next command is sent.

Command Time

Enter the maximum time allowed for each telnet command. If this command time has been exceeded, and the server has not replied back with a telnet command prompt, then the next command is sent.

Command\$

Enter your telnet commands, 1 per line. Always enter the exit command on the last line. This field also supports [dynamic variables](#). Using dynamic variables, you can enter a command, which will be dynamically resolved at runtime. example: cd /users/\$%USERNAME%\$

Example of Commands to enter

```
ls -lt
cd /user/test
./runTestScript.scr
exit
```

Custom Login

The Telnet(Adv) task always sends the username first, followed by the password. If your login method is different, than the Telnet(Adv) task will fail. In this case, please use the following procedure to apply a Custom login method. On the first line of the commands field, enter the String CUSTOMLOGIN^ (case important) followed by the number of commands you need for your custom login. Ex: CUSTOMLOGIN^3 should be the first line if you have 3 login commands. Then enter the rest of the commands like you normally would.

Example of Commands to enter for custom login

```
CUSTOMLOGIN^3
username
```

```
group
password
ls -lt
cd /user/test
./runTestScript.scr
exit
```

Unzip Archives

This task allows you to unzip files from a zip archive(s).

Source Directory

Enter the full path name of the directory, where your zip files exists. Note that the path is case sensitive for unix Ftp servers.

Filename\$

Enter the name filter for the zip archives you wish to extract, [using the following wildcard rules](#) . This field also supports [dynamic variables](#). Using dynamic variables, you can select a file based on current date/time.

Extract Directory

Enter the full path name of the directory, where you need to put the extracted files. Note that the path is case sensitive on unix systems.

Place files in new subdirectory

If you select "None", no new subdirectory is created, within the Extract Directory.

If you select Date, a new subdirectory based on the date, is created within the Extract Directory. All files are placed in this directory.

If you select Date, a new subdirectory based on date and time, is created within the Extract Directory. All files are placed in this directory.

Url Monitor

This feature enables you to monitor web pages for changes. It can work with multiple web sites. Each time this task runs, it updates a log file on your system, with the modified time stamp of each web page it finds, within the web site. If a change is detected, it will download the file to a selected directory. HTTPS downloads require additional SSL install files, which are available from our website.

Url to spider\$

Enter the full path of the URL's to monitor. You must include the protocol (i.e. http://). You can specify multiple URL's, by separating them with a comma. Example - http://www.hiteksoftware.com,http://www.espn.com. This field also supports [dynamic variables](#).

Hyperlink Depth

Select the Hyperlink depth(D) required. If D=0, only the root Url is monitored. If D=1, direct links (DL1) to the root Url are also monitored. If D=2, direct links(DL2) to each DL1 are also monitored, and so on....

WARNING: Use D=1 to test. If you have a slow connection, and/or the web site has many large files, this task could run for hours!!

Local Dir

Enter the full path of the local directory to place all the results. If a file is found to have been changed, then it is downloaded to this directory.

Filename Filter\$

Leave this field blank if you want all files to be downloaded, or enter the name filter for the files you wish to download [using the following wildcard rules](#) . This field also supports [dynamic variables](#).

Using dynamic variables, you can select a file based on current date/time.

You can also select only specific sub folder(s), to be downloaded, by entering the sub folders relative path. For the task to realize, that a sub folder filter is used, the filename filter has to include at least one path separator '/'. Example: /test or /test/subfolder etc..

You can use multiple filters by separating them with ^. Example: test.htm^prod would download test.htm, products.htm, produce.htm etc.. Please note that when you use multiple filters, only one of the filters has to be satisfied, for the url to be downloaded.

Maintain Timestamp

Select this option, if you need the downloaded file, to have the same timestamp, as the file on the web server.

Download Option

There is three options available:

- 1) Download always - This will overwrite the existing local file, irrespective of timestamp.
- 2) Download based on local file timestamp - If a local file exists in the download folder, a new file will be downloaded, only if the timestamp of the file on the web server, is newer than that of the existing local file.
- 3) Download based on archived url timestamp - Every time a file is downloaded by this task, its timestamp on the web server is saved locally. If this option is selected, a new file is downloaded from the web server, only if its timestamp is newer than the saved value.

Place files in new subdirectory

To regularly monitor certain URL's, you can save each download to a different directory, based on current time/date. Select the "Create New Subdirectory" option, which best suits your download frequency. A new subdirectory, based on current date or time, will be created. All downloaded files and results will be placed in this subdirectory.

Proxy Parameters

If you are behind a firewall, enter the proxy settings. Your network administrator can provide you with the proxy settings. You can also obtain the proxy settings from your web browser.

Variable Monitor

The Variable Monitor task is a special type of task which can be used in two ways:

- 1) If a variable criteria is met in this task, you can run another task.
- 2) The Variable Monitor task can also be embedded within a Chain task, and is useful for conditional task processing. If you want to use this task in a chain for conditional processing, select 'None' as the 'Task To Run'. If the criteria is satisfied, then an exit code = -100 is thrown. Else, an exit code = 0 is thrown.

Variable name

Select the variable you need to monitor from the list. For some variables like DATE, you need to manually enter the format.

Variable value

Enter the value of the variable you want to monitor. This value will be dependent on the comparison criteria you select.

Comparison criteria

Choose your comparison criteria. This will dictate how the actual value of the system variable is compared to the variable value you entered.

For **Integer** comparisons, use: =, !=, , < . The values found will be converted to integers and compared

For **String** comparisons, use: equals, contains, starts with, ends with

For **Date** comparisons, use: Older than or Newer than. For Date comparisons, the Variable Value should specify both the value and the field type (m=minutes, h=hours, d=days). Date comparison examples:

For '5 days' use: 5 d

For '3 hours' use: 3 h

For '30 minutes' use: 30 m

Task To Run

Select the desired task to run, if any of the selected criteria are met. If you want to use this task in a chain for conditional processing, select 'None'. You can also select multiple tasks to run in sequence or simultaneously. To run tasks in sequence, use taskTitle1|taskTitle2|taskTitle3. To run tasks simultaneously, use taskTitle1&taskTitle2&taskTitle3.

There are 4 types of variables listed:

Task Variables: Task variables values are generated/updated when the tasks run. Example: Tasks exit codes are stored as the following variable: TaskTitle::ExitCode

System Variables: Program system variables which are resolved when the current task runs, examples are DATE, USERNAME etc..

Java Variables: Java variables, which are available to the Java Virtual Machine. You can get the value of java variables from the Utilities menu / Java Variables

User Variables: Users can add their own variables via batch files, scripts or programs. Users can link output from their own programs, to this program.

Web Downloads

This feature enables you to download web pages, from your local intranet, or from the internet. The most common internet protocols `http://`, `https://`, & `ftp://` can be downloaded. This feature will work with most web pages, except web pages using frames. Please note that https downloads require additional SSL install files, which are available from our website.

Download Url \$

Enter the full path of the URL to download (`http://www.hiteksoftware.com`). **You must include the protocol (i.e. `http://`).**

This field also supports [dynamic variables](#). Using dynamic variables, you can select a URL based on current date.

Example: `http://www.test.com/$%pp-yy%$.htm`, will download `http://www.test.com/07-02.htm`

Many Ftp urls require username & password information to be sent in the url itself. The format may vary, but is usually of the form: `ftp://username:password@host.com/folder/file`

To hide your password, enter the password in the password field of the dialog box. Use the following format for the url:

`ftp://username:$password$@host.com/folder/file`

The string `$password$` will be replaced by the value, in the password field of the dialog box.

Examples:

- 1) `http://www.cnn.com`
- 2) `ftp://comp.com/test/test.dat`
- 3) `ftp://username:$password$@test.com/folder/test.zip`

Parameters (optional) \$

This is an optional field. If you did not enter input parameters in your url field, you can enter the parameter string here. You must separate each key=value pair by &

Example: `name=joe&age=25&date=2/12/01`

This field also supports [dynamic variables](#). Example: `name=$%USERNAME%$&age=25&date=$%DATE::pp/dd/yy%$`

Local Directory

Enter the full path of the local Directory to place the html file in.

Local Filename \$

Enter the required name of the downloaded file. Example `test.htm`. If you do not enter a name, a default

name of "TaskTitle.htm" is used. This field also supports [dynamic variables](#). Using dynamic variables, you can save the file to a customized name, based on date/time, or username, or other variable.

Maintain Timestamp

With this option selected, the download file will have the same timestamp, as the file on the web server.

Download Option

There are 3 download options:

- 1) Download always - This will overwrite the existing local file, irrespective of timestamp.
- 2) Download based on local file timestamp - If a local file exists in the download folder, a file will be downloaded, only if the file on the web server, is newer than the existing local file.
- 3) Download based on archived url timestamp - Every time a file is downloaded by this task, the file timestamp on the web server, is saved locally in a file. If this option is selected, a new file is downloaded from the web server, only if its timestamp on the web server, is newer than the saved value.

Append to Filename

If you are regularly downloading a certain URL and wish to save each download to a different filename [based on current time/date or custom code you set](#), choose the append file option which best suits your download frequency.

Proxy Parameters

If you are behind a firewall, enter the proxy settings. Your network administrator should be able to provide your proxy settings or you may also be able to obtain them from your web browser.

Download Embedded Images

This option allows you to download embedded images within the html page. The image formats supported are .jpg,.gif, .tif, .png, and .bmp. Only images in the same or child folder as the html page will be downloaded.

Windows Command

This task is only available on Windows systems. This task allows you to run executable programs(.exe) or batch files(.bat). The task launches a native windows application called WinCommand.exe. WinCommand.exe will actually run your application, or batch file. This Windows Command Task generally provides better results, for windows batch files, than the standard Command Task. Executable programs (.exe) should perform similarly using, either the Command Task, or the Windows Command Task.

Command Line \$

Enter the command line. If you want to launch an executable file, or batch file, enter the full path to the file. Example:

```
c:\windows\notepad.exe
```

If the path to your executable file, or batch file includes spaces, enter the path to the file in quotes.

Example:

```
"c:\program files\mybat.bat"
```

You can enter a commandline parameter, to pass to your executable file, or batch file. You should enter a space between the file path, and the parameter. Leave spaces between any subsequent parameters.

Example:

```
"c:\program files\mybat.bat" par1 par2
```

This field also supports [dynamic variables](#). For example, to launch a batch file, and pass it the current time, use:

```
c:\data\test.bat $%DATE::hh:mm:ss%$
```

Working Directory

The working directory is required for your program, or script, to run correctly. Use the full path to the Working Directory. You do not need to enclose the working directory in quotes, even if the path has spaces in it.

Environment Variables \$

Enter an array of strings. Each element of the array, defines a single environment variable setting. The format for each environment variable is name=value. You should separate each name=value pair, by a delimiter character. You can choose the delimiter to separate each element. The default delimiter is ^. If any of your variable name=value pairs contains the ^ character, then you should set another character, as your delimiter. The delimiter cannot be "=", or any other character, that appears in your name=value pairs.

Example of correctly formatted environment variables, on windows systems:

```
PATH=c:\test;c:\test2;c:\test3^TEMP=c:\temp
```

This field also supports [dynamic variables](#). For example, to launch a batch file and set the variable 'CUSTOM_DATE', use:

```
CUSTOM_DATE=%DATE::Qqq dd, YY HH:mm:ss am_pm%$
```

This will set the variable CUSTOM_DATE in the batch file to: Aug 21, 2002 10:21:23 AM etc..

Termination Time

The termination time is the maximum time that your program can run. The task will terminate your program, if it is still running, after the maximum allowed time. Enter a value = 0, if you do not want to terminate your program. In this case, the task will wait indefinitely, for your task to finish.

Polling interval

While your program is running, the task continuously polls your program. The task will check if your program has completed. It will obtain the Exit Code value, when the program exits. You can select the polling interval (default = 15 seconds). The minimum interval is 1 second. We recommend using 15 seconds.

Zip Directories

The Zip Directories task allows you to Zip an entire directory, and its sub-tree to a single zip archive. Each file in the directory is compressed, before being added to the zip archive.

Directory to Zip

Enter the full path name of the directory, where your source files exist. Note that the path is case sensitive for unix Ftp servers.

Filename\$

Enter the name filter for the files you wish to copy [using the following wildcard rules](#). This field also supports [dynamic variables](#). Using dynamic variables, you can select a file based on current date/time.

Include sub directories

Select this option, if you want to zip the entire sub directory tree, below the main directory.

Directory to place Zip Archive

Enter the full path name of the directory, where you need to put the newly created zip archive. Note that the path is case sensitive on unix systems.

Name for Zip Archive\$

Enter the filename for zip archive. This field also supports [dynamic variables](#). Using dynamic variables, you can save the file, to a customized name based on date/time, or username, or other variable.

Append to Filename

You can save each archive to a different filename, [based on current time/date, or on custom code you set](#).

Date Filter

If you need to filter files based on file modified date, select this option. For the between option, enter 2 values separated by '-' (2-4 etc..). Examples:

Older than 5 Minute, Newer than 2 Day, Between 3-5 Day, Between 1-4 Hour

Task Variables

Every task outputs values of certain variables. You can use these values in other tasks, or in the [Variable Monitor](#) task. You can view the list of Task Variables using the 'Utilities' Menu - 'Task Variables' menu item.

Using task variables in text fields

The variable must be formatted as: `%%TaskTitle::VariableName%%`

Example: for command task with task title = Test

`%%Test::ExitCode%%` = Exit code for latest run of this task

`%%Test::ErrorMsg%%` = Error message for latest run of this task

Task Type	Variable Name	Variable Description
All Tasks	LastTaskTitleRun	Task title of last task run
All Tasks	LastTaskTypeRun	Task Type of last task run
All Tasks	LastExitCode	Exit Code of last task run
All Tasks	TaskType::LastTaskRun	Task title of last task run for each task type
All Tasks	TaskTitle::ExitCode	Task exit code value
All Tasks	TaskTitle::LastRunEndTime	Time last run finished
All Tasks	TaskTitle::LastRunStartTime	Time last run started
All Tasks	TaskTitle::LastRunRunTime	Duration of last run
Command	TaskTitle::ErrorMsg	Task error message. This variable is BLANK unless an error is detected during the task run. This variable is set to BLANK before each run
Command	TaskTitle::OutputMsg	Task output message. This variable is BLANK unless output is detected during the task run. This variable is set to BLANK before each run
WinCommand	TaskTitle::ErrorMsg	Task error message. This variable is BLANK unless an error is detected during the task run. This variable is set to BLANK before each run
CopyFile	TaskTitle::TotalFiles	Number of files in directory
CopyFile	TaskTitle::PassFilename	Number of files passing filename filter check
CopyFile	TaskTitle::PassDate	Number of files passing date modified criteria
CopyFile	TaskTitle::ForceCopy	Number of older(or same) source files copied
CopyFile	TaskTitle::SourceOlder	Number of source files older (or same date) than target files (i.e. not copied)
CopyFile	TaskTitle::CopyFailed	Number of files failed to copy
CopyFile	TaskTitle::DeleteFailed	Number of files failed to delete after move/rename options
CopyFile	TaskTitle::FilesCopied	Number of files copied
DeleteFile	TaskTitle::TotalFiles	Number of files in directory
DeleteFile	TaskTitle::PassFilename	Number of files passing filename filter check
DeleteFile	TaskTitle::PassDate	Number of files passing date modified criteria
DeleteFile	TaskTitle::FilesDeleted	Number of files deleted
ZipDirs	TaskTitle::TotalFiles	Number of files in directory
ZipDirs	TaskTitle::PassFilename	Number of files passing filename filter check

ZipDirs	TaskTitle::PassDate	Number of files passing date modified criteria
ZipDirs	TaskTitle::FilesZipped	Number of files zipped
ZipDirs	TaskTitle::ZipFilePath	Zip file path
ZipDirs	TaskTitle::ZipFileSize	Zip file size
TextSearch	TaskTitle::FilesSearched	Number of files searched
TextSearch	TaskTitle::FilesFound	Number of files search term(s) were found in
TextSearch	TaskTitle::TotalFound	Total number of times search terms were found
TextSearch	TaskTitle::ResultsFile	Results filename
TextSearch	TaskTitle::TotalTermsFound	Total number of search terms found. This variable is useful for searches in single file. If you have X search terms, but only Y terms were found in 1 file, then the value of this variable = Y. But if, you have multiple files being searched, this value may be greater than the number of search terms.
TextSearch	TaskTitle::LastLineNumber	The last line number where a search term was found. Value is 1000000 if no search term is found.
Ftp	TaskTitle::TotalFiles	Number of files in directory
Ftp	TaskTitle::LocBackup	Total local files backed up
Ftp	TaskTitle::FailLocBackup	Total local files failed to back up
Ftp	TaskTitle::RemBackup	Total remote files backed up
Ftp	TaskTitle::FailRemBackup	Total remote files failed to back up
Ftp	TaskTitle::FilesDeleted	Total source files deleted
Ftp	TaskTitle::DeleteFailed	Total source files failed to delete
Ftp	TaskTitle::FailRename	Total temporary files failed to rename
Ftp	TaskTitle::PassFilename	Total files meeting filename criteria
Ftp	TaskTitle::PassDate	Total files meeting filename and date criteria
Ftp	TaskTitle::PassNew	Total files meetings newly modified files criteria
Ftp	TaskTitle::FailedXfers	Total failed transfers
Ftp	TaskTitle::FilesXfered	Total files transferred
FtpMonitor	TaskTitle::SourceDir	Remote Directory being monitored
FtpMonitor	TaskTitle::DirChanged	true if directory change is detected since last run, else false
FtpMonitor	TaskTitle::FileFound	Name of last file found that satisfies the monitoring criteria
FtpMonitor	TaskTitle::FirstFileFound	Name of first file found that satisfies the monitoring criteria
FtpMonitor	TaskTitle::FileSize	Size of file found that satisfies the monitoring criteria
FtpMonitor	TaskTitle::FileDate	Date of file found that satisfies the monitoring criteria
FtpMonitor	TaskTitle::TotalFiles	Total files in directory, if 'Total Files' criteria is selected
Ftp Monitor	TaskTitle::PassFileName	Number of files that pass the filename filter
Ftp Monitor	TaskTitle::Filenames	Files that pass monitoring criteria. The format is #file1^#file2... so that this variable can be directly used in other tasks filename field (example copy/ftp/delete etc..)
Ping	TaskTitle::LastResult	Last result for this task
Ping	TaskTitle::TestCount	Total test count
Ping	TaskTitle::UpCount	Total successfull connections
Ping	TaskTitle::UpPercent	Percent uptime
Ping	TaskTitle::LastRuntime	Last run time
Telnet	TaskTitle::Response1	The initial connection response from server + login prompt (All task variables are reset to BLANK before task runs)

Telnet	TaskTitle::Response2	password prompt
Telnet	TaskTitle::Response3	Authentication response/welcome message
Telnet	TaskTitle::Response4+	Response4, Response5 etc.. are the reply strings from the server for each command you send
Telnet(Adv)	TaskTitle::Response1	The initial connection response from server + login prompt (All task variables are reset to BLANK before task runs)
Telnet(Adv)	TaskTitle::Response2	password prompt
Telnet(Adv)	TaskTitle::Response3	Authentication response/welcome message
Telnet(Adv)	TaskTitle::Response4+	Response4, Response5 etc.. are the reply strings from the server for each command you send
EmailBulk	TaskTitle::Successful	Number of Successful emails
EmailBulk	TaskTitle::Failed	Number of Failed emails
EmailBulk	TaskTitle::Total	Number of Total emails
EmailCheck	TaskTitle::PassCrit1	Number of messages passing criteria 1
EmailCheck	TaskTitle::PassCrit2	Number of messages passing criteria 2
EmailCheck	TaskTitle::Messages	Number of messages being downloaded
EmailCheck	TaskTitle::Subject	Subject for the last email downloaded
EmailCheck	TaskTitle::From	Sender for the last email downloaded
Database SQL	TaskTitle::Rows	Number of rows returned or updated
File Monitor	TaskTitle::FileName	Filename for the last file that satisfied the monitoring criteria
File Monitor	TaskTitle::FirstFileName	Filename for the first file that satisfied the monitoring criteria
File Monitor	TaskTitle::FilePath	Filepath for the file that satisfied the monitoring criteria
File Monitor	TaskTitle::FileSize	File size in bytes for the file that satisfied the monitoring criteria
File Monitor	TaskTitle::FileDate	File timestamp for the file that satisfied the monitoring criteria
File Monitor	TaskTitle::TotalFiles	Total files in directory, if total files criteria is selected
File Monitor	TaskTitle::PassFileName	Number of files that pass the filename filter
File Monitor	TaskTitle::Filenames	Files that pass monitoring criteria. The format is #file1^#file2... so that this variable can be directly used in other tasks filename field (example copy/ftp/delete etc..)
Dir Monitor	TaskTitle::Filenames	Files that pass monitoring criteria. The format is #file1^#file2... so that this variable can be directly used in other tasks filename field (example copy/ftp/delete etc..)
Dir Monitor	TaskTitle::ModifiedObjects	Total modified objects (directories + files) in directory tree
Dir Monitor	TaskTitle::TotalObjects	Total objects (directories + files) in directory tree
Dir Monitor	TaskTitle::FileName	Last filename that passes monitoring criteria.
Dir Monitor	TaskTitle::FirstFileName	First filename that passes monitoring criteria.
Ftp Command	TaskTitle::ResponseX	Response1, Response2 etc.. are the reply strings from the ftp server for each command you send
Chain	TaskTitle::LastRunStepTitle	Latest step task title run in this chain task
Chain	TaskTitle::LastRunStepType	Latest step task type run in this chain task
Chain	TaskType::LastRunStepTitle	Latest step task title run in any chain task
Chain	TaskType::LastRunStepType	Latest step task type run in any chain task
Web	TaskTitle::FileSize	File size as reported by web server
Web	TaskTitle::TotalBytes	Total bytes downloaded
Web	TaskTitle::ContentType	File content type
Web	TaskTitle::ContentEncoding	File content encoding

Web	TaskTitle:: LastModifiedHeader	value of the last-modified header field
Web	TaskTitle:: DateHeader	value of the date header field
Web	TaskTitle:: ExpirationHeader	value of the expires header field
LogFile Monitor	TaskTitle::TotalAlerts	Total new alerts detected in this run
LogFile Monitor	TaskTitle::AlertLines	Variable contains all new alert lines. Use the File variable task to output lines to file.
LogFile Monitor	TaskTitle::LastAlertTime	Last alert time

System Variables

Many task fields support variable entries. This program has some coded system related variables, which you can use, in these task fields.

Using variables in text fields

The variable must be formatted as: `$$Variable_Name::Option1::Option2..%$`

Options may only be required by some Variables.

examples: `$$DATE::hh:mm;yyyy%$` or `$$USERNAME%$`

The following is the list of System Variables

1) DATE

You can specify the current date using the following format

`$$DATE::dd/pp/yy hh:mm:ss AMPM%$`

You can also choose any date, time, or date/time separators like /, _, -, or : etc...

YY = 4 digit year will be output (2001 etc..)

yy = 2 digit year will be output (00-99)

pp = 2 digit month will be output (01-12)

ppx = 1 or 2 digit month will be output(1-12)

qqq = 3 character month will be output (jan-dec)

QQQ = 3 character month will be output (JAN-DEC)

Qqq = 3 character month will be output (Jan-Dec)

dd = 2 digit date will be output (01-31)

ddx = 1 or 2 digit date will be output(1-31)

HH = 2 digit hour based on 12 hour clock will be output (01-12) (should generally be used with am_pm)

HHx = 1 or 2 digit hour based on 12 hour clock will be output(1-12)

hh = 2 digit hour based on 24 hour clock will be output (00-23)

hhx = 1 or 2 digit hour based on 24 hour clock will be output (0-23)

mm = 2 digit minute will be output (00-59)

mmx = 1 or 2 digit minute will be output (0-59)

ss = 2 digit seconds will be output (00-59)

am_pm = AM is output for AM hours, PM is output for PM hours

wy = 2 digit week of the year

wyx = 1 or 2 digit week of the year

dy = 3 digit day of the year

dyx = 1,2, or 3 digit day of the year

dw = day of the week (1-7) (country dependent, ex: 1 = sunday in the US, and 1 = monday in France)

wm = week of the month (1-5)
dwm = day of week in the month (1-5)

Examples: (for date/time = June 21st, 2005, 3:45:30 PM)

`%%DATE::dd-pp-YY%%$ = 21-06-2005`

`%%DATE::dd-pp1-yy%%$ = 21-6-05`

`%%DATE::qqq/dd/yy_hh:mm:ss%%$ = jun/21/05_15:45:30`

`%%DATE::Qqq dd, YY, HH:mm am_pm%%$ = Jun 21, 2005, 03:45 PM`

2) DATEADD

This variable returns the value of a single time field (date, hour, minute etc..), after adding (or subtracting) days/hours etc.. from the current time:

`%%DATEADD::dd::X%%$`

where X is the number to add. For next day, use 1, for previous day/hr etc.. use -1.

Examples: (for date/time = June 21st, 2005, 3:45:30 PM)

`%%DATEADD::dd::1%%$ = 22`

`%%DATEADD::dd::-1%%$ = 20`

`%%DATEADD::hh::1%%$ = 16`

`%%DATEADD::hh::-1%%$ = 14`

`%%DATEADD::mm::1%%$ = 46`

`%%DATEADD::mm::-1%%$ = 44`

Notes:

1) You can only add or subtract and receive one field at a time

2) Only numerical date / time values are supported. i.e. days of week (mon, tue etc..) or months (jan, feb etc..) are not supported.

3) You cannot embed a DATEADD variable within a DATE variable. You have to use all variables in sequence.

Example: If you want current month and date, but require previous hour, in the format month-date_hr, you would use

`%%DATE::pp-dd_%%$%%DATEADD::hh::-1%%$ = 06-21_14`

2A) DATEADDX

This variable returns a formatted date/time string, after adding (or subtracting) days/hours etc.. from the current time:

`%%DATEADDX::Format::dd::X%%$`

where X is the number to add. For next day/hour etc., use 1, for previous day/hour etc.. use -1. Format is the date/time format that you need the result to be returned in.

This variable allows user to add date/hour/minute to current time intelligently. Ex: adding a day to 12/31/03, rolls over to 01/01/04, not to 12/32/03

Examples: (for date/time = June 21st, 2005, 3:45:30 PM)

`$(DATEADDX::YY-pp-dd::dd::1%)$ = 2004-06-22`

`$(DATEADDX::YY-pp-dd_hh-mm::hh::-1%)$ = 2004-06-21_14-45`

`$(DATEADDX::YY-pp-dd_hh-mm::hh::1%)$ = 2004-06-21_16-45`

`$(DATEADDX::YY-pp-dd_hh-mm::mm::1%)$ = 2004-06-21_16-46`

`$(DATEADDX::YY-pp-dd_hh-mm::mm::-1%)$ = 2004-06-21_16-44`

Notes:

1) You can only add or subtract one field at a time

2) Only numerical date / time values are supported. i.e. days of week (mon, tue etc..) or months (jan, feb etc..) are not supported.

3) DATEMONTH

This variable allows you to get the last day of the previous month, or first Sunday of next month etc.. It returns a formatted date/time string. The format required is:

`$(DATEMONTH::format::month_offset::day_val::X%)$`

format is the date/time format that you need the result to be returned in

month_offset = 0 for current month, 1 for next month, -1 for previous month

day_val = sun, mon, tue, wed, thu, fri, sat to specify day of week. Use 'day' to specify date of the month.

X = week of month (1-5) or day of month (1-31). To specify the last week of the month, use -1. To

specify the last date of the month, use -1.

Examples: (for month = April, 2004)

To get first Sunday of this month:

`$(DATEMONTH::YY-Qqq-dd::0::sun::1%)$ = 2004-Apr-04`

To get last Sunday of this month:

`$(DATEMONTH::YY-Qqq-dd::0::sun::-1%)$ = 2004-Apr-25`

To get first Monday of next month:

`$(DATEMONTH::YY-Qqq-dd::1::mon::1%)$ = 2004-May-03`

To get last Monday of previous month:

`$(DATEMONTH::YY-Qqq-dd::-1::mon::-1%)$ = 2004-Mar-29`

To get last day of previous month:

`$(DATEMONTH::YY-Qqq-dd::-1::day::-1%)$ = 2004-Mar-31`

4) FILE

This variable outputs information on file size, date or can parse lines from the file into the variable. This is useful if you need to monitor files for certain strings etc.. This variable can be used in the 'File Variable' task, to process text in files, and output results to new files.

`$(FILE::SIZE::[file_path]%)$`

`$(FILE::DATE::[format]::[file_path]%)$`

`$(FILE::LINE::FIRST::[file_path]%)$`

`%%FILE::LINE::LAST::[file_path]%%$`
`%%FILE::LINE::MID::[start_line]::[end_line]::[file_path]%%$`
`%%FILE::LINECONTAINS::FIRST::[string]::[file_path]%%$`
`%%FILE::LINECONTAINS::LAST::[string]::[file_path]%%$`
`%%FILE::LINECONTAINS::ALL::[string]::[file_path]%%$`
`%%FILE::LINENUMCONTAINS::FIRST::[string]::[file_path]%%$`
`%%FILE::LINENUMCONTAINS::LAST::[string]::[file_path]%%$`
`%%FILE::LINESAFTER::FIRST/LAST::[string]::[offset]::[file_path]%%$`
`%%FILE::LINESBEFORE::FIRST/LAST::[string]::[offset]::[file_path]%%$`

Examples:

1) `%%FILE::SIZE::c:\temp\test.txt%%$`
will output the size of the file in bytes

2) `%%FILE::DATE::qqq-dd-yy_hh:mm:ss::c:\temp\test.txt%%$`
will output the last modified time of the file, in the date/time format you specify. Please see the **DATE** variable above, for details on how to format the date. example: if June 21,2002, 3:45 pm was the last modified time of the file c:\temp\test.txt, this variable will output: jun-21-02_15:45:30

3) `%%FILE::LINE::FIRST::c:\temp\test.txt%%$`
will output the first line in the file c:\temp\test.txt

4) `%%FILE::LINE::LAST::c:\temp\test.txt%%$`
will output the last line in the file c:\temp\test.txt

4) `%%FILE::LINE::MID::5::90::c:\temp\test.txt%%$`
will output the lines 5-90 in the file c:\temp\test.txt. If there are only 65 lines in the file, it will output lines 5-65.

6) `%%FILE::LINECONTAINS::FIRST::error alert::c:\temp\test.txt%%$`
will output the first line in test.txt, that contains the string 'error alert'. It will return an empty string, if the 'error alert' is not found.

7) `%%FILE::LINECONTAINS::LAST::error alert::c:\temp\test.txt%%$`
will output the last line in test.txt, that contains the string 'error alert'. It will return an empty string, if the 'error alert' is not found.

8) `%%FILE::LINECONTAINS::ALL::error alert::c:\temp\test.txt%%$`
will output all the lines in test.txt, that contains the string 'error alert'. It will return an empty string, if the 'error alert' is not found in any line.

9) `%%FILE::LINENUMCONTAINS::FIRST::error alert::c:\temp\test.txt%%$`

This variable will output the line number of the first line, that contains the string 'error alert'. It will return 1000000, if the string is not found.

10) `%%FILE::LINENUMCONTAINS::LAST::error alert::c:\temp\test.txt%`

This variable will output the line number of the last line, that contains the string 'error alert'. It will return 1000000, if the string is not found.

11) `%%FILE::LINESAFTER::FIRST/LAST::[string]::[offset]::[file_path]%`

This variable will output all lines, after the line that contains the String [string] .

Use FIRST to output all lines, after the first line containing the [string]

Use LAST to output all lines, after the last line containing the [string]

use [offset] = 0 , to include the line containing the [string], use [offset] = 1, to start at next line, etc..

examples:

`%%FILE::LINESAFTER::FIRST::error::0::c:\temp\test.txt%` (will include the line containing the string 'error')

`%%FILE::LINESAFTER::LAST::error::1::c:\temp\test.txt%` (will not include the line containing the string 'error')

12) `%%FILE::LINESBEFORE::FIRST/LAST::[string]::[offset]::[file_path]%`

This variable will output all lines, before the line that contains the String [string] .

Use FIRST to output all lines, before the first line containing the [string]

Use LAST to output all lines, before the last line containing the [string]

use [offset] = 0 , to include the line containing the [string], use [offset] = 1, to end at previous line, etc..

examples:

`%%FILE::LINESBEFORE::FIRST::error::0::c:\temp\test.txt%` (will include the line containing the string 'error')

`%%FILE::LINESBEFORE::LAST::error::1::c:\temp\test.txt%` (will not include the line containing the string 'error')

5) Other Variables

USERNAME

This variable outputs the value of the current user

example: if your username is 'john'

`%%USERNAME%` =john

USERHOME

This variable outputs the value of the current users home directory

example: if your username is 'john', your home directory on win 2000 would be

`%%USERHOME%` =c:\documents and settings\users\john

TEMPDIR

This variable outputs the current value of the temp folder

`%TEMPDIR%` =c:\temp

OSNAME

This variable outputs name of the operating system

example: on Windows 2000

`%OSNAME%` =Windows 2000

OSVERSION

This variable outputs the version of the operating system

example: on Windows 2000

`%OSVERSION%` =5.0

IPADDRESS

This variable outputs the computers IP address

example: if your systems IP address is 192.168.0.1

`%IPADDRESS%` =192.168.0.1

SYSNAME

This variable outputs the computer name on the network

example: if your computer name is Win1

`%SYSNAME%` =Win1

User Defined Variables



This feature provides the commandline code to set an existing variable, add a new variable, or get the value of a variable. You can use this code, from an external program, script, batch file, or commandline (command prompt in windows or terminal in unix). The commandline and working directory required to set the variable, will be automatically generated in the text fields. You can use the "Copy to Clipboard" feature to copy these fields to clipboard for use in your scripts or programs.

A) Setting your own variables for the program (Automize, AbleFtp etc..) to recognize

From the commandline, via your batch files / scripts etc, first set the program install folder as the current directory

```
cd install_folder (i.e. cd ..\Automize or cd ..\AbleFtp etc..)
```

on windows, use:

```
jre\bin\java.exe -cp .;inputs;jclasses.jar UserVariable set variable_name variable_value
```

IMPORTANT: variable_name cannot include any spaces in it

on Unix/MacOSX, use:

```
java -cp .:inputs;jclasses.jar UserVariable set name value
```

where variable_name = variable name, and variable_value = variable value

B) Getting User variable values for use in your scripts or batch files

From the commandline, via your batch files / scripts etc, first set the program install folder as the current directory

```
cd install_folder
```

on windows, use:

```
jre\bin\java.exe -cp .;inputs;jclasses.jar UserVariable get variable_name
```

```
jre\bin\java.exe -cp .;inputs;jclasses.jar UserVariable get variable_name c:\test\test.txt (outputs the variable to the file test.txt)
```

IMPORTANT: variable_name cannot include any spaces in it

on Unix/MacOSX, use:

```
java -cp ./inputs:jclasses.jar UserVariable get variable_name
```

```
java -cp ./inputs:jclasses.jar UserVariable get variable_name /home/test.txt (will output the variable to the file test.txt)
```

where variable_name = variable name, and variable_value = variable value

Return Value:

The variable value will be returned as a string value in the system output stream. You should add code in your batch, script, or program, to read this value.

Also, if the variable is an integer, the exit code of the command above, is the integer value of the variable.

If the variable is not an integer, the exit code of the command above = -1001.

How To's

[1. How to send email notification if task fails](#)

[2. How to send email notification depending on whether task fails or passes](#)

1. How to send email notification if task fails

Generally, if your task exit code = 0 or <0, then the task has passed. If exit code is 0, then task generally has failed. Use the Email Notification feature, from the Settings menu, to email yourself if a task fails.

2. How to send email notification depending on whether task fails or passes

Generally, if your task exit code is 0 or <0, then the task has passed. If exit code is 0, then task generally has failed.

1. Setup your desired task and test it.
2. Setup two email tasks to yourself. One with the failure notification message, and another with success notification message. Test your email tasks.
3. Setup a chain task with 3 steps.
 - 1st step should be your desired task.
 - 2nd step in chain should be the email failure notification.
 - 3rd step should be email success notification.
4. For your desired task in the 1st step, in the chain options, set the criteria:
If Exit Code < 1, Go to Step 3.
5. For the email failure notification task in step 2, Select the Check Box "Always stop chain at this Step".
6. If your task passes, the exit code is 0 or less, the chain will move to Step 2. A success notification email will be sent, and chain will stop here.
6. If your task fails, the exit code is 0. The chain will move to step 3, and failure notification email will be sent.

Windows specific tutorials

[Windows Batch Files](#)

[Windows Batch File Examples](#)

[Backing up using XCopy on Windows](#)

Windows Batch Files

Using Batch Files is a very powerful way to enhance your productivity.

Batch files are simple text files with commands.

Each command has to be entered on a new line followed by a carriage return.

The Text file is saved as 'xxxx.bat' to generate a batch file. . The maximum length of the filename is 8 characters.

Create a Simple Batch file

Open up a new text document using notepad.

Type in the Following command on the first line: Start notepad

From the 'File Menu' select 'Save As'

In the File Save Dialog Box, select 'All Files (*.*)' as the File Type.

Enter the name of the batch file to save as. Ex: Test.bat

Running a Batch file

To run a batch file, double click on its icon. For example, the Simple batch file code above, would launch notepad.

Passing Command line parameters to a batch file

The parameters are entered after the full path to the batch file. There is a space between parameters.

Example: c:\mybat.bat command1 command2 command3

Using Command line parameters in a batch file

Within the batch file, use %1 to refer to the first command line parameter (%2 for the second parameter etc.). This parameter is passed while launching the batch file.

Ex: To launch a batch file, and tell it to copy the file "c:\test.txt" to drive a:, we would do the following:

1) Launch your batch file(c:\mybat.bat) using: c:\mybat.bat c:\test.txt

2) In mybat.bat, use the following code: Copy %1 a:

Concept of Start/Current directory

If you plan to use relative paths in your batch file, it should know its current directory.

For example, to delete all text files in the current directory, we would use a batch file with the following

code: DEL *.txt

This would work, if you run the batch file, by clicking on its icon.

However, if you used another batch file, located in a different directory, to run this batch file, then this code will not work. You need to enter the full path: DEL c:\junk*.txt

Batch File Examples

In the examples below, Filepath1, Filepath2 refer to the full path of the executable files.

Ex: c:\program files\programs.exe or c:\windows\notepad.exe

The syntax for the Start command varies between Win95 and WinNT families.

Example 1: Starting programs simultaneously

Start Filepath1

Start Filepath2

Example 2: Starting 2nd program after 1st finishes

Start /w Filepath1

Start /w Filepath2

The /w switch tells the windows to wait for the programs to finish.

Example 3: Running program with command line arguments

Start /w Filepath1 Commandline

You need to enter the commandline arguments after the Filepath1. A blank space is required after the Filepath1 and the first commandline argument. A blank space is also required between commandline arguments if you need to specify more than one argument.

Useful DOS Commands for Batch Files

From a MS-DOS window, type in 'command /?' for details on the commands. Some of these commands may not be available to you, depending on your system. For some commands like Ftp, type 'ftp' in a dos window. At the subsequent prompt, type 'help'.

Commands which may be of use to you

attrib, break, call, command, copy, chcp, chdir (cd), chkdsk, cls, date, debug, del, deltree, dir, diskcopy, doskey, defrag, drvspace, dir, echo, edit, erase, exit, expand, fc, find, for, format, fdisk, ftp, goto, if, label, mem, mkdir (md), mode, more, move, netstat, net, path, pause, ping, prompt, rem, rename (ren), rmdir (rd), set, shift, sort, start, subst, time, type, verify, vol, xcopy

Backing up using XCopy

MS-DOS comes with a xcopy utility, which you can use, to backup files, or directories. The exact

syntax varies with operating systems. Win 95, Win NT, and Win 2000 pro, have different syntax for XCopy. From a command window use `xcopy /?` for more information.

For example, to backup the `c:\junk` directory and all its sub directories to `a:`, and replace older files with newer files, and overwrite existing files without prompting, use
`xcopy c:\junk a: /D /E /Y` (the `/Y` switch may or may not be used depending on your system)

[Back to Index](#)

Introduction

The CommandLine Module (CLM) allows a user to:

- 1) Control the scheduler engine on a remote system
- 2) Allow a programmer to control the scheduler tasks and schedules from their own programs, scripts or batch files

The CLM is an optional addon to our regular programs. It is a 30 day free trial and requires a separate license and registration code. The CLM is installed along with the main software.

Using the CommandLine Module

You can control the scheduler engine using the Commandline Module (CLM) via Telnet (or rlogin, ssh etc..).

Windows:

1. Open a DOS command window.
2. Telnet to the remote system where the program is installed
3. 'cd' to the main install folder (Example: cd "c:\program files\hiteksoftware\automize" or cd "c:\program files\Automize")
4. The main install folder includes the following file: clm.bat
5. Use following command to verify that clm is setup correctly: clm ? . If clm is setup correctly, you will get a listing of all commands.

NOTE: You may get the following error: "The system cannot find the path specified". This means that you do not have the Java machine installed into the main install folder. You probably used the zip installer and are using a 'java' machine which is in your system path. In this case you will have to edit 'clm.bat' and replace the commandline:

```
jre\bin\java -cp .;jclasses.jar;inputs CLM %*
```

with:

```
java -cp .;jclasses.jar;inputs CLM %*
```

Unix & MacOSX:

1. Open a Terminal window.
2. Telnet to the remote system where the program is installed
3. 'cd' to the main install folder
4. The main install folder includes the following file: clm
5. Run the command: chmod +x clm
6. Use following command to verify that clm is setup correctly: ./clm ? . If clm is setup correctly, you will get a listing of all commands.
7. You can add the appropriate shell header line to the file 'clm' to avoid having to type in './' everytime you run a command

NOTE: You may get a "command not found" error. This means that you do not have 'java' in your system path. You probably used the Install including 'JVM' and have the Java machine (jre folder) installed into the main install folder. In this case, you will have to edit 'clm' and replace the commandline.

```
java -cp .:jclasses.jar:inputs CLM $*
```

with:

```
jre/bin/java -cp .:jclasses.jar:inputs CLM $*
```


[Back to Index](#)

Usage: clm command[::par1][::par2]...

Command List

? - Displays this command reference page

?::add - Help information on how to add a new task or modify a task

?::addx - Help information on how to add a new schedule or modify a schedule

?::pars - Help information on task parameters 1-50 for each task type

?::type - Help information on TaskTypes used

?::user - Help information on various user settings

init::program_name - program_name = Automize, AbleFtp, or JaSFtp etc (Case sensitive). This command is only required if you are using the clm for the first time, and also have not used the user interface yet

start - Starts scheduler engine

stop - Stops scheduler engine

status - Checks if scheduler engine is running

list - Displays Task List

list::n - Displays task data for task 'n' in a format suitable for editing

listx - Displays Schedule List

list::n - Displays schedule data for task 'n' in a format suitable for editing

run::n - runs task number 'n' from task list.

run::taskTitle - runs 'taskTitle' from task list.

add::taskType|*|taskTitle|*|Par1|*|Par2|*|...|*|... - adds a new task to task list

addx::taskType|*|taskTitle|*|NextRun|*|...|*|... - adds a new schedule to schedule list

mod::n::taskType|*|taskTitle|*|Par1|*|Par2|*|...|*|..... - modifies task data for task 'n'

modx::n::taskType|*|taskTitle|*|NextRun|*|...|*|... - modifies schedule data for schedule 'n'

del::n - deletes task number 'n' from task list

delx::n - deletes schedule number 'n' from schedule list

sch::n - schedules task 'n' in schedule list

sus::n - suspends task 'n' in schedule list

out - Displays output log

act - Displays activity log

err - Displays debug log

task::taskTitle - Displays the Tasklog for taskTitle

listUser - Displays a list of user settings

getUser::name - returns the value of user 'setting'

setUser::name::value - sets the user setting value

listVar - Displays a list of user variables

getVar::variable_name - returns the value of user variable

setVar::value_to_set::variable_name - sets the user variable value

syncList - Displays a list of synchronization profiles

syncDis::profile - Displays the settings for the synchronization 'profile'

syncAdd::profile::taskList::option::waitTime::afterWait - Adds a new profile or modifies synchronization profile data if profile exists.

syncDel::profile - Deletes the synchronization profile

emailList - Displays a list of email server profiles

emailDis::profile - Displays the smtp server settings for the email 'profile'

emailAdd::profile::host::port::emailAddress::senderName::authenticationType::popServer::user::pwd - Adds a new profile or modifies email profile data if profile exists.

emailDel::profile - Deletes the email profile

emailNotifyList - Displays a list of email notification profiles

emailNotifyDis::profile - Displays the email notification settings for this profile

emailNotifyAdd::profile::true/false::notificationEmail::"crit"::exitCodeValue - Adds a new email notification profile or modifies profile data if profile exists. Example: to send notification email if task exits with exit code 0, use: clm emailNotifyAdd::true::me@test.com::"="::0

emailNotifyDel::profile - Deletes this email notification profile

backupDir - Displays the last backup directory

backup::dir_path - backup data to dir_path

backup::default - backup data to last backup directory

restore::dir_path - restores data from dir_path

restore::default - restores data from last backup directory

registerSched::program::code - Registers & unlocks the scheduler engine for specific 'program' using the registration 'code'. program = Automize or AbleFtp or JaSFtp etc..

registerCLM::code - Registers & unlocks this CommandLine Module using 'code'

NOTE: n is 0 based (i.e. first task in task or schedule list has index = 0)
More CLM help information is available in the user interface help file.

[Back to Index](#)

Modifying a Task

1. Get a listing of all tasks using: `clm list`
2. Display the desired task 'n' in a format suitable for listing using: `clm list::n`
3. Usage: `clm "mod::n::taskType|*|taskTitle|*|Par1|*|Par2|*|...|*|Par10....."`
4. NOTE: Task list starts at index 0. If you have 10 tasks, first task is 0 and last task is 9
5. [taskTypes and parameters can be obtained using: `clm ?::pars`](#)

Adding a Task

Usage: `add::taskType|*|taskTitle|*|Par1|*|Par2|*|...|*|Par10....`

Another option is to duplicate an existing task using: `clm copy::n` and then modify the parameters.

Par1-Par50

These parameters are dependent on the task. These parameters may also change with versions, if more features are added to a task. These parameters are exactly in the same sequence in the task table of the user interface. If all 50 parameters are not used, they should be left blank. **However, correct number of |*| are required.**

You can also Manage your tasks on a remote machine using the user interface itself in conjunction with this CLM and an ftp client. See the help files on CLM for details.

[Back to Index](#)

Task parameters - updated for Version 6.x

These parameters are dependent on the task. These parameters may also change with versions, if more features are added to a task. These parameters are exactly in the same sequence in the task table of the user interface. If all 50 parameters are not used, they should be left blank. **However, the correct number of |*| are required when used in the CLM.**

Command and Windows Command

Parameter1 = commandline

Parameter2 = working directory

Parameter3 = polling interval in seconds

Parameter4 = terminate time in minutes

Parameter5 = environment parameters

Parameter6 = environment parameters delimiter

Copy Files

Parameter1 = source directory

Parameter2 = target directory

Parameter3 = filename

Parameter4 = append to filename (None, Date , Date/Time or custom code)

Parameter5 = include subdirectory option? (true/false)

Parameter6 = create new subdirectory (None, Date, Date/Time)

Parameter7 = date based copy? (true/false)

Parameter8 = older than/newer than

Parameter9 = time (integer number)

Parameter10 = Minute/Hour/Day

Parameter11 = ignore file timestamp? (true/false)

Parameter12 = rename filename

Parameter13 = copy/move/rename

Zip Directories

Parameter1 = source directory

Parameter2 = target directory

Parameter3 = filename

Parameter4 = append to filename (None, Date , Date/Time or custom code)

Parameter5 = include subdirectory option? (true/false)

Parameter6 = create new subdirectory (No, Date, Date/Time)

Parameter7 = date based zip? (true/false)

Parameter8 = older than/newer than

Parameter9 = time (integer number)

Parameter10 = Minute/Hour/Day

Parameter11 = path type

Delete Files

Parameter1 = source directory

Parameter2 = filename

Parameter3 = date based delete?(true/false)

Parameter4 = older than/newer than

Parameter5 = time (integer number)

Parameter6 = Minute/Hour/Day

Parameter7 = delete subdirectories(true/false)

Parameter8 = delete empty subdirectories(true/false)

Text Search

Parameter1 = source directory

Parameter2 = filename

Parameter3 = search term(s)

Parameter4 = search term separator

Parameter5 = surrounding lines to append to results file

Parameter6 = case sensitive search? (true/false)

Parameter7 = results directory

Parameter8 = results filename

Parameter9 = append to results filename (None, Date , Date/Time or custom code)

Parameter10 = replace? (true/false)

Parameter11 = replace term(s)

Parameter12 = include directories (true/false)

Ping

Parameter1 = host

Parameter2 = port

Telnet

Parameter1 = host

Parameter2 = port

Parameter3 = delay time between commands in seconds

Parameter4 = commands separated by *EOL* (Example: username*EOL*password*EOL*ls -lt*EOL*exit)

Telnet(Adv)

Parameter1 = host

Parameter2 = port

Parameter3 = username

Parameter4 = password
Parameter5 = maximum login time (seconds)
Parameter6 = login prompt (typically :)
Parameter7 = maximum command time (minutes)
Parameter8 = polling interval for each command (seconds)
Parameter9 = telnet prompt (\$, % or etc..)
Parameter10 = prompt characters, default = 10
Parameter11 = commands separated by *EOL* (Example: username*EOL*password*EOL*ls -
lt*EOL*exit)

Ftp

Parameter1 = remote directory
Parameter2 = local directory
Parameter3 = filename (based on wildcards, see help for details)
Parameter4 = Ascii/Binary
Parameter5 = ftp option (Put File(s)/Get File(s)/Delete File(s)
Parameter6 = append to filename (None, Date or Date/Time)
Parameter7 = log transfer? (true/false)
Parameter8 = transfer modified files only? (true/false)
Parameter9 = include subdirectories? (true/false)
Parameter10 = ftp profile name
Parameter11 = transfer based on modified date? (true/false)
Parameter12 = date value
Parameter13 = delete source file after transfer? (true/false)
Parameter14 = transfer with temporary extension? (true/false)
Parameter15 = temporary extension (ex: .tmp)
Parameter16 = backup local file? (true/false)
Parameter17 = local backup directory
Parameter18 = backup remote file? (true/false)
Parameter19 = remote backup directory
Parameter20 = maintain timestamp (true/false)
Parameter21 = transfer modified source files using Ftp log? (true/false)
Parameter22 = comparison criteria (newer than/ older than)
Parameter23 = period type (Day/Hour/Minute)

Ftp Monitor

Parameter1 = remote directory
Parameter2 = filename
Parameter3 = file exists check (true/false)
Parameter4 = file length check (true/false)
Parameter5 = ftp length option (greater than/less than)
Parameter6 = file length value in kb
Parameter7 = file date check (true/false)

Parameter8 = file date option (newer than/older than)
Parameter9 = file date value
Parameter10 = file date option (minute/hour/day)
Parameter11 = file totals check (true/false)
Parameter12 = file totals option (greater than/less than)
Parameter13 = file totals value
Parameter14 = Run task if any change is detected (true/false)
Parameter15 = Task to run (tasktitle)
Parameter16 = Ftp profile name

Ftp Commands

Parameter1 = host
Parameter2 = port
Parameter3 = password
Parameter4 = commands (Each Command should be separated by *EOL*, Ex: user test*EOL*pass
mypwd*EOL*PWD)

Email

Parameter1 = recipients (format = To:email1,[Cc:email2,Bcc:email3,Reply-To:email4]
Parameter2 = subject
Parameter3 = attachments (format = filepath1,filepath2)
Parameter4 = message (newlines separated by *EOL*, Ex: this is a *EOL* new line)
Parameter5 = email profile name
Parameter6 = append to attachment (None, Date or Date/Time)
Parameter7 = headers

Bulk Email

Parameter1 = recipient list file path
Parameter2 = subject
Parameter3 = attachments (format = filepath1,filepath2)
Parameter4 = message (newlines separated by *EOL*, Ex: this is a *EOL* new line)
Parameter5 = email profile name
Parameter6 = append to attachment (None, Date or Date/Time)
Parameter7 = delay
Parameter8 = headers

Check Email

Parameter1 = criteria1 (date/sender/subject)
Parameter2 = option1 (newer than X days/older than X days/newer than X hours/older than X hours/
equals/contains/starts with/ends with)
Parameter3 = value1
Parameter4 = criteria2 (date/sender/subject)
Parameter5 = option2 (newer than X days/older than X days/newer than X hours/older than X hours/

equals/contains/starts with/ends with)

Parameter6 = value2

Parameter7 = and/or

Parameter8 = mail folder path

Parameter9 = append to filename (None, Date or Date/Time)

Parameter10 = server type (POP3/IMAP)

Parameter11 = host

Parameter12 = port

Parameter13 = user

Parameter14 = password

Parameter15 = delete message

Web

Parameter1 = download URL

Parameter2 = local directory

Parameter3 = filename

Parameter4 = append to filename (None, Date or Date/Time)

Parameter5 = authentication required (true/false)

Parameter6 = host

Parameter7 = port

Parameter8 = user

Parameter9 = password

Parameter10 = download images? (true/false)

Parameter11 = parameters

Parameter12 = file date check (true/false)

Parameter13 = file date criteria (greater than / less than etc..)

Parameter14 = date value

Parameter15 = date type (minute/hour/day)

Parameter16 = maintain timestamp (true/false)

Parameter17 = download option (Download always/Download based on local file timestamp/ Download based on archived url timestamp)

Url Monitor

Parameter1 = url to monitor

Parameter2 = spider depth

Parameter3 = local directory

Parameter4 = place files in subdirectory (No, Date or Date/Time)

Parameter5 = authentication required (true/false)

Parameter6 = host

Parameter7 = port

Parameter8 = user

Parameter9 = password

Parameter10 = filename

Parameter11 = download option
Parameter12 = maintain timestamp (true/false)

Database SQL

Parameter1 = sql statement
Parameter2 = results directory
Parameter3 = results filename
Parameter4 = append to filename (None, Date or Date/Time)
Parameter5 = driver class name
Parameter6 = database url
Parameter7 = authentication required (true/false)
Parameter8 = username
Parameter9 = password

File Monitor

Parameter1 = directory
Parameter2 = filename filter
Parameter3 = file exists? (true/false)
Parameter4 = file length check? (true/false)
Parameter5 = greater than/less than
Parameter6 = size in kb (integer number)
Parameter7 = file date check? (true/false)
Parameter8 = older than/newer than
Parameter9 = time (integer number)
Parameter10 = Minute/Hour/Day
Parameter11 = total files check? (true/false)
Parameter12 = greater than/less than
Parameter13 = number of files (integer number)
Parameter14 = Task to Run (must be valid taskname in task table)

Directory Monitor

Parameter1 = Directory to monitor
Parameter2 = Filename filter
Parameter3 = include subdirectories (true/false)
Parameter4 = Task to Run (must be valid taskname in task table)

Directory Change

Parameter1 = Directory to monitor
Parameter2 = maximum wait time in minutes (integer number)
Parameter3 = polling interval in seconds (integer number)
Parameter4 = Task to Run (must be valid taskname in task table)
Parameter5 = Use as daemon (true/false)

Variable Monitor

Parameter1 = Variable name

Parameter2 = Variable value

Parameter3 = comparison criteria (equals, contains, starts with, ends with, "=", "!=", "", "<")

Parameter4 = Task to Run (must be valid taskname in task table)

File Variable

Parameter1 = Variable name

Parameter2 = Filepath to output variable to. (ex: c:\data\test.txt)

Parameter3 = Filing options (insert at start or append at line . etc..)

Parameter4 = Option value corresponding to Filing Option chosen

Parameter5 = Add line separator before inserting variable

Parameter6 = Add line separator after inserting variable

Archive Logs

Parameter1 = archive directory

Parameter2 = append to filename (None, Date or Date/Time)

Parameter3 = place files in subdirectory (No, Date or Date/Time)

Auto Backup

Parameter1 = archive directory

Parameter2 = not used

Parameter3 = place files in subdirectory (No, Date or Date/Time)

Email Logs

Parameter1 = email addresss to receive notification (separate multiple addresses by comma ,)

Chain

Parameter1 = total steps in chain

Parameter2 = All the Chain data in a single line

Data for each step is separated by |@|

Each step has multiple entries separated by |#|

Entry1 = task type

Entry2 = taskTitle

Entry3 = maximum wait time in minutes

Entry4 = Skip to step exit code choice (true/false)

Entry5 = exit code comparison criteria (= , != , , <)

Entry6 = exit code value

Entry7 = Skip to step (step number or taskTitle)

Entry8 = rerun step? (true/false)

Entry9 = rerun exit code comparison criteria (= , != , , <)

Entry10 = rerun exit code value

Entry11 = number of times to rerun step
Entry12 = delay time between reruns
Entry13 = Always stop chain at this step?

Wakeup

Parameter1 = Mouse Move/Key Stroke/Mouse Click

Shutdown

Parameter1 = Logoff / Reboot / Shutdown / Poweroff

Parameter2 = Win 95/98/ME / Win NT/2000/XP

Open Documents

Parameter1 = Document path or name etc..

MSAccess

Parameter1 = Full path to MSAccess.exe

Parameter2 = Database path

Parameter3 = Option (0 = none/autoexec macro , 1 = run macro, 2 = compact database, 3 = repair database)

Parameter4 = Macro name

Parameter5 = terminate time in minutes

MSExcel

Parameter1 = Full path to Excel.exe

Parameter2 = Workbook path

Parameter3 = Option 0/1 (0 = none/autoexec macro , 1 = run macro)

Parameter4 = Macro name

MSWord

Parameter1 = Full path to winword.exe

Parameter2 = Database path

Parameter3 = Option 0/1 (0 = none/autoexec macro , 1 = run macro)

Parameter4 = Macro name

Parameter5 = terminate time in minutes

MSPowerpoint

Parameter1 = Full path to ppowerpnt.exe

Parameter2 = Presentation path

Parameter3 = Option 0/1/2 (0 = none/autoexec macro , 1 = print presentation, 2 = run slideshow)

Parameter4 = terminate time in minutes

[Back to Index](#)

Modifying a Schedule

1. Get a listing of all tasks using: `clm listx`
2. Display the desired task 'n' in a format suitable for listing using: `clm listx::n`
3. Usage: `clm "mod::n::Scheduled|*|taskType|*|taskTitle|*|NextRun|*|Frequency|*|Period|*|Hour|*|Minute|*|Details|*" "`
4. NOTE: Schedule list starts at index 0. If you have 10 Scheduled tasks, first Schedule is 0 and last Schedule is 9

Adding a Schedule

Usage: `clm "addx::Scheduled|*|taskType|*|taskTitle|*|NextRun|*|Frequency|*|Period|*|Hour|*|Minute|*|Details|*" "`

`taskType` & `taskTitle` are case and space sensitive

`Scheduled` = true/false (true = scheduled, false = suspended)

`taskType` = [Enter task type exactly. To see taskType list, use: `clm ?::type`](#)

`taskTitle` = enter title exactly as it is listed in task table

`NextRun` = DO NOT MODIFY!!! user interface use only

`Frequency` = 1-100

`Period` = Second/Minute/Hour/Day/Week/Month

`Hour` = 0-23 (only needed when scheduling by Day/Week/Month)

`Minute` = 0-59 (generally 0)

`Details` = Monday, Tuesday, Wednesday, Thursday, Friday, Saturday,

Sunday, 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,

The run hours 0-23 in the Details are only required when scheduling by Second, Minute or Hour.

You can also Manage your tasks on a remote machine using the user interface itself in conjunction with this CLM and an ftp client. See the help files on CLM for details.

Remote Install

Some users need to install the software onto remote computers, which may be miles away. This is also useful for IT administrators who need to install on multiple systems quickly and monitor and control multiple systems. Others may need the commandline module because their system does not have display capabilities or only serve as servers.

If you need to install remotely, you can simply copy over the entire main install folder from your Local system to the Remote system using Ftp (or simple Copy if the remote folder is accesable as a shared drive to you). Then access the CLM using Telnet (or rlogin, ssh etc..) to start/stop and perform other scheduler operations. NOTE: when you install remotely, the normal system shortcuts are not created on the remote systems desktop or other folder since the installation software is not used.

The program does not require any system wide properties to be set. On Windows, nothing is read/written to the registry or system folders/files etc... On Unix systems, the system classpath or path is not set or accessed. Hence, only a simple copy or Ftp of the entire main install folder to the Remote System is required.

Steps for Remote Install:

- 1) Install a master copy of the software on the same platform as the target remote systems. i.e. if you want to perform remote installs on Windows systems, Install a master copy on a Windows system.
- 2) Ftp the entire main install folder to the target remote systems.

[Back to Index](#)

Using Ftp to set/view remote data

The easiest way to setup or view data on a remote system is via the user interface itself on your local system. [See the section on managing multiple remote schedulers](#). **All the data files are in the 'inputs' folder within the main install folder.**

Setting (export) data to remote system

1. Telnet to and Shut down the scheduler on the remote system using: `clm stop`
2. Launch the user interface to create the tasks and schedules on your local system itself.
3. Use an ftp client to transfer all files in the 'inputs' folder from your local machine to the 'inputs' folder on the remote machine
4. On the remote system, start up the scheduler using: `clm start`

View (import) data from remote system

1. **BACKUP your local scheduler data, using the Backup feature, if the scheduler is also running on your local system!!!**
2. Shut down the scheduler on your local system.
2. Shut down the user interface on your local system.
3. Use an ftp client to transfer all files in the 'inputs' folder from the remote machine to the 'inputs' folder on your local machine.
4. Start up user interface on your local system. Shut down the scheduler engine so that it does not run any tasks on your system.
5. Make any changes that you desire. Then you can export the data back to the remote system (see section above).
6. Restore your local scheduler data, using the Restore feature, if the scheduler is also running tasks on your local system

Managing remote scheduler(s)

We developed the commandline module on a Windows system and tested it by controlling scheduler engines running on MacOSX, Linux and Solaris. You can manage remote scheduler(s) either by:

1. [Using only the CLM](#)
2. Using the CLM in combination with the user interface & Ftp

Managing Remote Systems using the CLM in combination with Local User Interface & Ftp

The easiest way to setup or view data on a remote system is via the user interface itself on your local system. To prevent confusion, it helps to understand and organize the data well on your local system to prevent confusion between the programs data on your local system and the remote system.

1. The main install directory contains an 'inputs' folder
2. All the data files are in this 'inputs' folder. The user interface saves all your updates to data files in this folder.
3. Create new folders in the folder called 'inputs_remote1', 'inputs_remote2', 'inputs_remote3' etc.. for each remote system that you will control. In our case, we used 'inputs_mac', 'inputs_solaris', and 'inputs_linux'.
4. Create a new folder in the main install folder called 'inputs_localhost' for your local system, if you also run tasks on your local system.
5. If the scheduler is also running tasks on your local system, shut down the scheduler & shut down the user interface. **Then Backup (copy) all the data from the 'inputs' folder to 'inputs_localhost'**
6. If you wish to view data from 'remote2', you can either copy the data from 'inputs_remote2' to the 'inputs' folder or [by ftp directly FROM the 'inputs' folder on the 'remote2' system.](#)
7. Launch the user interface to create the tasks and schedules for 'remote2'
8. After you have modified the data for 'remote2', you can backup the 'inputs' folder to 'inputs_remote2' and [ftp directly TO the 'inputs' folder on the remote system.](#)
9. Then use the CLM module to STOP and then START the engine on 'remote2'. This will force the engine to load the new data.
10. If the scheduler is also running tasks on your local system, you can restore your local systems program data from 'inputs_localhost' to 'inputs'.

Programmatically controlling the scheduler

There are 2 ways to control the Automize scheduler engine from your programs

1) **Need to run Automize tasks only.** Use the TaskRunner feature from the Automize user interface to generate the commandline code. Then use this code in your batch files, unix scripts, java exec function, c/c++ exec/spawn functions etc... to run the tasks. The task will return an exit code that can be used in your programs for conditional processing. Please see the help file on Exit Codes for details.

2) **Full control of engine via commandline module:** Adding, modifying, suspending the tasks/schedules, start/stop the engine, or run tasks, and many more features are available via the commandline module. You would have to format and send the desired commands via the commandline module using your program. Please use the commandline module manually to see how it works, before trying to program through it.

Scheduling a task to run on the first Monday of the month.

Description

User needs to schedule a task (TaskTitle = TaskX) to run on the first Monday of the month, at 6:00 AM. Automize scheduler currently does not support this type of schedule. However, this schedule can be obtained by using a Variable monitor task. This solution is possible in version 5.3 and later.

Solution

1) Create an extra Variable monitor task using the following parameters:

Variable Name = `$(DATE::dwm%)$`

Variable value = 1

Comparison Type = equals or =

Task to Run = TaskX (i.e. task that needs to run on first Monday of month)

2) Schedule the Variable monitor task using the following parameters:

Frequency = 1

Type = Week

Date/time = 6:00 AM

Day = Monday

The first Monday of the month should always lie between dates 1-7. The Day in Week of Month (dwm) = 1, for dates 1-7. i.e. the first Monday of the month should have dwm = 1.

The Variable monitor task will be triggered every Monday at 6:00 AM. It will then check the value of the Variable 'dwm'. If dwm = 1, it will trigger the task TaskX to run. If dwm > 1, it will not trigger TaskX to run. Hence the task TaskX will be run only on the first Monday of the month.

Monitoring a website - receive alert when website is down

Description

User needs to constantly monitor his website and gets informed by email to his cell phone when his website is down. He uses ATT and can receive emails to his cell phone. Email address is: phone_num@att.mobile.net. Most cell phone companies now allow users to receive email to their cell phones. Please see your cell phone manual if this feature is available for your cell phone.

The ping task also keeps statistics. So user can check the uptime statistics for his website from the Logs / TaskLogs menus. The statistics are updated in a file called "TaskTitle_statistics".

Solution

1) Create a Ping task (http port = 80) which checks if the website is up

Host = Ip address (not website url)

example: Host = 192.168.0.1, Port = 80

2) Create an Email task which sends email to your cell phone email address. Make sure that you enter a small message. Most cell phones cannot accept messages greater than ~ 150 characters.

3) Create a chain task.

Step 1 = Ping task

Step 2 = Email task

Set the Chain Exit Code option for the Ping Task as follows:

If Exit Code = 0, Go to step 100 (i.e. stop chain)

4) Schedule this chain to run every 10 minutes.

If the website is up, for the ping task an Exit Code = 0 is returned, and the chain stops (step 100).

If the website is down, an Exit Code = 1 is returned. In this case the Email task is run and you will receive an email alert to your cell phone.

Monitoring a website using reference - receive alert when website is down

Date Updated - 06/03/2005

Description

This example is used by us (Hitek Software) to monitor HitekSoftware.com. HitekSoftware.com website is hosted by Interland.com. Normally, our website is up >99.9% of the time. However, we need to be informed by email to our cell phones when our website is down. We use AT&T and can receive emails to our cell phones. Email address is: phone_num@att.mobile.net. Most cell phone companies now allow users to receive email to their cell phones. Please see your cell phone manual if this feature is available for your cell phone.

Sometimes the internet connection to our company is down. i.e. our Internet Service Provider is down. This leads to false alerts being generated by the ping task which monitors our website. Hence, we use a reference website. Our ping task is only run if the reference website is up.

The ping task also keeps statistics. So we check the uptime statistics for HitekSoftware.com from the Logs / TaskLogs menus. The statistics are updated in a file called "TaskTitle_statistics".

Solution

1) Create a Ping task (http port = 80) which checks if the reference website is up

Host = cnn.com, Port = 80

2) Create a Ping task (http port = 80) which checks if our website is up

Host = hiteksoftware.com, Port = 80

3) Create an Email task which sends email to your cell phone email address. Make sure that you enter a small message. Example: 'hiteksoftware.com is down'. Most cell phones cannot accept email messages greater than ~ 150 characters.

4) Create a chain task.

Step 1 = Ping task (CNN)

settings: If Exit Code > 0, Go to step 100 (step 100 => stop chain)

i.e. stop chain if reference is down => our internet connection is down

Step 2 = Ping task (HitekSoftware)

settings: If Exit Code = 0, Go to step 100

i.e. stop chain and do not send alert email since our website is up

Step 3 = Email task (default settings)

4) We schedule this chain to run every 10 minutes.

If the hiteksoftware.com website is up, for the ping task, an Exit Code = 0 is returned, and the chain stops (goes to step 100).

If the hiteksoftware.com website is down, an Exit Code = 1 is returned for the ping task (step 2). In this

case the Email task is run, and we receive an email alert to our cell phones. If we receive 2-3 consecutive alerts (down for 20 minutes) we contact Interland.com and inform them that our website is down.

Monitor a log file for alerts. Receives immediate cell phone/pager notification.

Date Updated - 06/03/2005

Description

User is an equipment engineer in a manufacturing setting. He uses Automize to detect alerts/errors in his production equipments log file.

The format of his log file is as follows:

Aug 05, 2002, 14:45:53 - Alert - low chamber pressure....

Aug 05, 2002, 14:45:57 -

The alerts are always output with the following word 'Alert'. He needs to be notified immediately of any alerts that occur in the last 1 hour

Solution

You should be using version 5.3 or later

1) Create a TextSearch task (TaskTitle = scanlog)

Directory = directory where log file resides

Filename = log file name

Results file = alert.txt (results file folder = c:\alerts)

Search terms = \$%DATE::Qqq dd, YY, hh%\$

Surrounding lines to append = 0

This task will look for all lines which contain the current date and time, and have been logged in the current hour. Only lines which have been logged in the current hour will be output to the file alert.txt.

2) Create an email task (TaskTitle = emailAlert) to send yourself an email to your Cell phone/pager number. Most cell phone companies in the US provide the ability to receive emails to cell phone.

3) Create a Variable Monitor task (TaskTitle = AlertMonitor)

Variable Name = FILE::LINECONTAINS::ALL::Alert::c:\alerts\alert.txt

Variable Value = Alert

Comparison Criteria = contains

Task To Run = emailPager

If the file alert.txt contains the word 'Alert' then it means that an alert has been generated in the last 1 hour. Then the emailAlert task is run to send an alert to the users cell phone.

4) Create a Chain task as follows:

Step 1 = TextSearch Task (scanLog)

Step 2 = Variable Monitor Task (AlertMonitor)

5) Schedule the Chain to run every 1 hour, at 59 minutes after the hour.

Note that this technique will not detect alerts that are generated in the 59th minute of every hour. The user configured his equipment so that it is doing its 3 minute required maintenance during the minutes 58-01, so he does not miss any alerts

Monitoring Automize log files for errors / exit codes

Date Updated - 06/03/2005

Description

User needs to have the Automize output log clear to 0 bytes (i.e. delete the log) rather than truncate the log by half when the maximum size threshold is reached. This is required because the user is using an enterprise level Alarming software, to monitor the Automize logs for exit codes and alarms, when a chain fails,. They had a problem with false alarms with their monitoring software. When the log is truncated, the monitoring software would lose it's pointer and would start looking for alarms from the beginning of the log. This caused false alarms.

Solution

1) In Automize, use the logs menu and set the maximum log sizes to be extremely large (~1 MB). This way this limit is never reached, and the automatic truncation to 50% will not occur.

2) Create a delete task which deletes the Automize output log file. The output log is called 'outputLog.jsd'. It is in the Automize\inputs folder. Other task specific logs are in the Automize\TaskLogs folder.

Delete Task settings:

Title = delete_task

Directory =Automize\inputs

Filename = outputLog.jsd

3) Create a File Monitor task with following settings:

Directory =Automize\inputs

Filename = outputLog.jsd

Task to Run = delete_task

Select Option: Run task if file size greater than 200 KB

4) Schedule the File Monitor task to run at a frequency which ensures that the log file is deleted using the delete task, rather than automatically truncated to 50% by Automize.

Detect errors in telnet tasks using variable monitor

Date Updated - 06/03/2005

Description

The Telnet tasks can only send the telnet commands and get replies. The telnet task cannot detect if the command failed or if the batch file generated errors.

User is running a batch file on a Windows server using the telnet task. He needs to detect when errors are generated in his batch file.

Solution

1) When the Telnet task is run, every response from the Telnet server is stored as a Variable in Automize. Please see the help section on Task Variables for details. The variable name is of the form: task_title::ResponseX, where X is the response number and task_title is the telnet task title

2) When the users batch file is run, the Telnet sends back its data reply to Automize. This reply is stored in the variable task_title::Response4. If an error occurs in the batch file the word 'error' is output back to Automize and stored in the Response variable..

3) Create a **Variable Monitor** task, using the following settings:

Variable Name = task_title::Response4

Variable Value = error

Comparison criteria = contains

Task To Run = Email (send alert email when his batch file fails)

4) Create a Chain task with the following steps:

step 1 = Telnet task

step 2 = Variable Monitor task

The telnet task will generate/update the value of the variable task_title::Response4. The variable monitor task will check if the variable value contains the word 'error'. If it does, it sends an email to the user.

Save results of a telnet command to local file

Date Updated - 06/03/2005

Description

User needs to logon to telnet server, send a command, and save the results of his command to a local file

Solution

- 1) Create a Telnet or Telnet(Adv) task. When the Telnet task is run, every response from the Telnet server is stored as a Variable in Automize. Please see the help section on Task Variables for details. The variable name is of the form: task_title::ResponseX, where X is the response number and task_title is the telnet task title
- 2) After sending the users telnet command, Automize receives data from the server. This data is stored in the variable task_title::Response4.
- 3) Create a **File Variable** task, using the following settings:
Variable Name = task_title::Response4
Full Filepath = c:\data\telnet_data.txt
Filing Options = Create new file or overwrite existing file
Option Value = [leave empty]
- 4) Create a Chain task with the following steps:
step 1 = Telnet task
step 2 = File Variable task

The telnet task will generate/update the value of the variable task_title::Response4. The File Variable task will write the results out to telnet_data.txt.

Daemon to monitor directory for changes and run file processing program

Date Updated - 06/03/2005

Description

User is an IT professional in a production environment. He needs to monitor a folder for incoming files. Whenever a file appears in this folder, he needs to launch a script that will process the files data.

Solution

This solution is available with patch update 5.3_1 or later (release date = 09/15/02)

1) Create the Command / WinCommand task to launch your file processing script (TaskTitle = fileCrunch)

2) Create a Dir Change task as follows:

Dir to monitor = c:\production\incoming

Task to Run = fileCrunch

Polling interval = 15 seconds

Maximum time allowed for task = 1440 (1440 minutes = 1 day)

Select the Daemon checkbox

3) Schedule this task to run every day (i.e. 1440 minutes) i.e. this task will run continuously 24 hours a day. If a new file comes into the folder the file processing program is launched via the Command / WinCommand task.

File Monitor task to monitor directory for total accumulated files

Date Updated - 06/03/2005

Description

User needs to monitor a folder for incoming files. Whenever the total files in the folder exceeds 10, he needs to launch a script that will process the files files and move them out of the folder.

Solution

1) Create the Command / WinCommand task to launch your file processing script (TaskTitle = fileCrunch)

2) Create a File Monitor task as follows:

Dir to monitor = c:\production\incoming

Filename = [leave blank]

Task to Run = fileCrunch

Select the option: Run task if total files in directory greater than 10

3) Schedule this task to run every 2 minutes i.e. this task will run continuously 24 hours a day. If more than 10 files are accumulated into the folder, the file processing program is launched via the Command / WinCommand task.

Run task if file exists in a directory, else send an email

Date Updated - 06/03/2005

Description

I have a need to send an email, if a file does not exist in a directory. If the file does exist, I need a macro to run.

Solution

1) Create a File Monitor using the following settings:

Dir = dir_name

File = filename

Select option = Run task if file exists

Task to Run = Your_Macro_task

2) Create a Chain Task with the following settings:

Step 1 = File Monitor task

Options = If Exit Code = -100, Go to step 100

Step 2 = Your_Email_Task

Schedule this Chain task. If the file exists in the folder, then the File Monitor task will run the Macro Task. In this case, the File Monitor task returns with an exit code = -100. Hence, in the Chain task will exit (i.e. go to step 100) without sending the email.

If the file does not exist in the folder, then the File Monitor task will not run the Macro Task, and it will return with an exit code = 0. In this case, step 2 of your chain will run, i.e. the Email task

Dynamically download URLs that are specified in a text file

Date Updated - 06/03/2005

Description

User question: I have an application that must download different URL's. Can your software download a URL specified in a small text file that changes say every few minutes?

Solution

The text file 'c:\test\url.txt' should specify on the first line, the URL to download. Use Automize system variable 'FILE::.....' to dynamically read in the first line of this file.

Create a Web task using the following settings:

Download URL = `;%FILE::LINE::FIRST::c:\test\url.txt%`

Below is the output of such a test:

Sep 16, 2002 2:12:23 PM Web - web

Sep 16, 2002 2:12:23 PM Parsing variables: Input=`;%FILE::LINE::FIRST::c:\test\url.txt%`

Sep 16, 2002 2:12:23 PM Finished parsing variables: Result=`http://www.cnn.com`

Sep 16, 2002 2:12:26 PM

Finished downloading: `http://www.cnn.com`

Downloaded filename = `C:\test\web.htm`

Downloaded file size in bytes = 41658

Sep 16, 2002 2:12:27 PM Web - web - Exit Code = 0

Delete files from Ftp server directory, that are older than 30 days

Date Updated - 06/03/2005

Description

User question: How to delete files that are older than 30 days from my Ftp server

Solution

This is only possible using version 5.3_1 and later. The Ftp task has the following option, which is normally used with the Ftp Put/Get options:

Option = Transfer files if date modified is Older/Newer than X Days/Hours/Minutes.

Select the Ftp Delete option and select the option:

Option = Transfer if Date modified in Older than 30 Days.

This will delete files from the Ftp folder entered, that have not been modified in the last 30 days.

When new files come into folder, send email and attach these files

Date Updated - 06/03/2005

Description

User question: When new files are added to a directory, I would like to automatically create and send an email with those files as attachments. Also, the files need to be moved out to a backup folder after the email has been sent.

Solution

You would have to create 5 tasks: File Monitor task, Email task, Copy task, Delete task, Chain task

1) File Monitor task -

Dir to monitor = c:\data_folder

Filename = (leave blank. This will monitor all files)

Task To Run = None

Options = Run task if File exists

2) Email task

In the attachment field, use

[DIR]c:\data_folder[FILE]*

This will attach all files from the c:\data_folder to the email

3) Copy task

Source Dir = c:\data_folder

target Dir = c:\backup_folder

Filename = (leave blank - this will backup all files to the c:\backup_folder)

4) Delete task

Source Dir = c:\data_folder

Filename (leave blank - this will delete all files in the c:\data_folder)

5) Chain task (4 steps)

Step 1 - File monitor task

Select Option : If exit code = 0, Go to Step 100. This makes sure that if no file(s) are found, then the email, copy and delete tasks are not run. If file(s) are found then exit code = -100, and steps 2-4 in chain will be run

Step 2 - Email task - default options

Step 3 - Copy task - default options

Step 4 - delete task - default options

6) Schedule the chain task to run at your desired frequency.

Monitor email server: ensure it is processing incoming and outgoing emails

Description

User question: We need to make sure that our email server is processing internal and external emails correctly at all times. I need the software to send an email out to a remote mail server every five minutes. The remote mail server is configured to forward the test messages back to a test account in our internal email server. I need the software to also retrieve email from the test account every ten minutes. If no messages have been received for more than ten minutes, I need the software to print out an error document to printer.

Solution

You would have to create 4 tasks: Email Send task, Check Email task, Print task, Chain task

1) Email Send task

Send test email to external server

2) Email Check task

Criteria 1= download messages if date is newer than 10 minutes.

If any email is found matching the specified criteria (i.e. newer than 10 minutes), then the message is downloaded and an exit code = -100 is set in the task. If no new email within last 5 minutes is found on the server, than exit code = 0 is set.

3) Print task

Prints error document to the default printer

4) Chain Task

step 1 = email send task

step 2 = check email task, step 2 option: if exit code = -100, go to step 100 (i.e. do not print error document)

step 3 = print task, (this print task would run, only if step 2 returned with exit code = 0, i.e. no new email is found in last 10 minutes)

5) Schedule this chain task to run every 5 minutes

Backup a file and receive email notification of success/failure

Date Updated - 06/03/2005

Description

User question: I need to backup (copy) a file, if it exists. I need to send a command successful confirmation email when the backup is completed successfully and a failure notification when the backup fails. (user is using a Windows 2000 system)

Solution

1) Run a copy command via a WinCommand task, using the windows NT command shell (cmd.exe). This way the system exit code is returned correctly to Automize when the copy fails.

In a WinCommand task:

For the commandline use: `cmd.exe /c copy DHR.html DHRCopy.html`

For the Working directory use: `c:\temp`

If the file exists and copy is successful, the WinCommand task will return with exit code = 0. If file does not exist, exit code = 1.

2) Create a Chain task with 3 steps

step 1 = WinCommand task from above

option = If exit code >0, go to step 3 (failure email)

step 2 = success email task

option = Chain always stops at this step

Step 3 = failure email task

Zip each file in a folder into its own archive

Date Updated - 06/03/2005

Description

User question: I would like to zip each file into its own archive. For example a folder contains 100's of .eps files each with a unique name of course. Example: B0123400ab.eps

I would like to zip this into its own archive assuming the name of the original file with a .zip extension.

Example: B0123400ab.zip

I then need to have the original file deleted from the source directory.

I then use the ftp task to send them to my customer.

Solution

1) Create an extra ZIP folder

2) Create a FileMonitor task:

task Title = fileMonitor

Dir to Monitor = your DATA folder (which has the .eps files)

Filename = (leave blank)

Criteria = Run task if file size > 0

Task To Run = None

3) Create a Zip task:

taskTitle = zipTask

Source Dir = DATA folder

filename = \$%fileMonitor::FileName%\$

Dir to place zip archive = ZIP folder

zip filename = \$%fileMonitor::FileName%\$.zip

Your zip filename will be of the form: B0123400ab.eps.zip .There is no way around this.

4) Create a Delete task:

taskTitle = deleteTask

Source Dir = DATA folder

filename = #\$\$\$fileMonitor::FileName%\$

(the # is to ensure that an exact filename match is done)

5) Create a Chain:

Step 1 = fileMonitor task

option: if exit code = 0, go to step 100 (i.e. if no files exist, stop chain)

Step 2 = Zip task

option: if exit code > 0, go to step 100 (i.e do not run step 3 if zip fails)

Step 3 = Delete task

6) You can schedule this chain to run every 1 minute or so (probably in a 2-4 hour range at night). Every time the chain runs, if a file is found, it will be zipped to the ZIP folder and deleted from the DATA folder.

7) You can then run a ftp task to ftp all files from the ZIP folder to your client. You can also clean up old files from the ZIP folder using a delete task which will delete all files from the ZIP folder.

Download webpages and append them to master file

Date Updated - 06/03/2005

Description

I have a task that visits a url containing a text file every 10mins and it saves it in a directory with the DATE and TIME format as its name...what I need to do is setup a task that will go to that directory say every 60mins, merge only the files with todays DATE and TIME and save the results to another file say for example results.txt

Solution

Thanks for evaluating Automize. You would have to insert/append the new downloaded file to a BASE file using the File Variable task. Additionally, you would have to download the webpages (textfiles) to a staging folder which contains the BASE file. After the new file is inserted/appended into the BASE file, you would have to move the webpage (textfile) to the final storage folder.

To do this, you would have to create the following tasks:

1) your web task

Download folder = [Staging folder] which contains BASE file.

2) A File Monitor task (Title = FileMon). This simply gets the name of the last downloaded file.

Directory = [Staging folder]

Filename = none

Task To Run = NONE

Select Option = Run Task if file exists

The FileMonitor task will now contain the variable `;%FileMon::FileName%` which is the name of the last downloaded web page.

3) a File Variable task to append file to the BASE file

variable name = `;%FILE::LINE::MID::0::10000::full_BASE_file_path%`

File Path = `[Staging Folder]\;%FileMon::FileName%`

Select Filing options as needed

4) a Copy task - This moves the file from the staging folder to a final folder. This is required for the FileMonitor task to work correctly in the next scheduled run.

Option = Move

Source Folder = [Staging Folder]

Target Folder = [Final Folder]

Filename = `;%FileMon::FileName%`

5) a chain task (use big blue links icon on front panel).

Step 1 = Web Task

Step 2 = File Monitor task

Step 3 = File Variable task

Step 4 = Copy task

6) Instead of scheduling the web task to run every X minutes, schedule this chain every X minutes.


Ftp Log

The ftp log maintains a list of the files that have been transferred. The exact nature of what is logged, depends on the options you chose, when you create an Ftp task. If the logging option is selected, the task will continuously update the log with every file transfer.

Managing Log Size

You can set the maximum Ftp log size. The Ftp log has a default maximum size of 100,000. This allows about 700-900 rows. The FTP log is automatically trimmed back to 50%, once the maximum log size is exceeded.

Saving Log to Disk

Enter a valid directory on your system, in the directory text box. Hit the save button . The file will be saved to the selected directory, with the filename 'FtpLog'.

[Back to Index](#)

TaskTypes

The taskType field is case sensitive as well as space sensitive.
For Version 6.x, the following are correct taskTypes entries:

Command
Windows Command
Alarm
Message
Wakeup
Open Documents
Shutdown
MSAccess
MSExcel
MSWord
MSPowerpoint
Macro
Copy Files
Zip Directories
Delete Files
Print Files
Text Search
Ping
Ftp
Ftp Monitor
Ftp Commands
Send Email
Bulk Email
Check Email
Web
Url Monitor
Database SQL
Maintenance
Archive Logs
Auto Backup
Email Logs
Chain
Dir Monitor
Dir Change

Variable Monitor

File Monitor

File Variable

Telnet

Telnet(Adv)